ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD CARLOS III DE MADRID

GRADO EN INGENIERÍA EN TECNOLOGÍAS
DE TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO DE FIN DE GRADO

DISEÑO Y DESARROLLO EN LAMP DE UN JUEGO EDUCATIVO DE PREGUNTAS Y RESPUESTAS

Autor: Ignacio Ortiz Luengo
Tutora: Iria Estévez Ayres
Marzo 2015

Agradecimientos.

En esta sección quería agradecer públicamente a las personas que han hecho que este largo camino sea posible.

Y, sobre todo, la paciencia de mis padres, el ánimo de mis hermanos, la ayuda y el apoyo de mis amigos (esos telecos!). Gracias David, gracias Eva, gracias Ana.

Gracias Iria por ser no solo una gran profesora, si no una de las que me han marcado. Ha sido un gran placer trabajar contigo.

Índice general

Índice de figuras.

Acrónimos y siglas

HTML: Hyper Text Markup Language
LOPD: Ley Orgánica de Protección de Datos
RGBD: Registro de Protección de Datos
AEPD: Agencia Española de protección de datos.
PHP: HyperText Preprocessor
HTTP: HyperText Transfer Protocol
IIS: Internet Information Services

Capitulo 1:Introducción.

En esta introducción se procederá a explicar la motivación para la realización de este proyecto. También se señalarán los objetivos propuestos y el marco regulatorio que afecta a la aplicación web desarrollada. Al final de la sección se dará una pequeña explicación de cada capitulo para instruir al lector en la estructura general del documento.

1.1. Motivación.

Desde la llegada del plan Bolonia la mayor parte de las asignaturas de las carreras universitarias se han convertido en cursos con grandes volúmenes de trabajo semanal que muchas veces son arduos en su realización lo que provoca que muchos de los alumnos abandonen las asignaturas al poco de empezar las clases con su correspondiente evaluación continua.

Dado que casi todas las asignaturas tienen una parte (más o menos grande) teórica, se ha considerado que la creación de un pequeño juego de preguntas de múltiples respuestas sería una buena herramienta tanto para alumnos y profesores.

Para los alumnos sería una herramienta de competición que sirve de varias formas. Primero para motivar el estudio de una asignatura. Esta motivación se consigue a lo largo del curso, realizando una o varias competiciones por tema que generen una dinámica de aprendizaje distinta a la convencional. Segundo para comprobar el nivel los exámenes y el nivel de los conocimientos que se va a requerir en la evaluación de la asignatura

Para los profesores de las asignaturas puede servir también de varias formas. Además de para motivar a los alumnos rompiendo la dinámica de las clases tradicionales con una actividad distinta, puede servir para ver si se están adquiriendo, en general, los conocimientos esperados, para hacer hincapié en los temas que más dificultad generen a los alumnos con más preguntas, o incluso utilizando la aplicación (como se explicará más adelante) como herramienta para realizar exámenes.

Este proyecto puede servir, además, para que otros alumnos generen, en el futuro, nuevos modos de competición dentro de la arquitectura que se propondrá a continuación y/o para que lo exporten a una aplicación móvil.

1.2. Objetivos.

Teniendo en cuenta lo descrito en el apartado anterior, se ha procedido al diseño e implementación de una aplicación web con una serie de requisitos y que sirva para los profesores de la universidad como aplicación para realizar competiciones de preguntas tipo test.

Para este propósito se han planteado los siguientes objetivos:

- Realizar un análisis teórico de las distintas tecnologías con las que se puede realizar el proyecto.
- Realizar un análisis del marco regulador.
- Diseñar e implementar las funcionalidades necesarias de la aplicación web.
- Validar el correcto funcionamiento de la aplicación.
- Crear una base solida sobre la cual poder realizar mejoras en el futuro y exportarlo a otros formatos.

1.3. Marco regulatorio.

Al tratarse de una programa que utiliza y guarda información personal de un conjunto de usuarios lo primero que hay que investigar es el marco regulador. En España tenemos que centrarnos en la ley orgánica de protección de datos (LOPD) y en el organismo que vela por el cumplimiento de dicha ley: la agencia española de protección de datos (AEPD). [1]

La LOPD es la que estipula las obligaciones que deben cumplir los desarrolladores y administradores del programa en cuanto a los datos almacenados se refiere. Esta ley afecta tanto a organizaciones publicas como privadas.

Según esta ley, la responsabilidad de la conservación de los datos recae sobre la persona, órgano administrativo o entidad que decide sobre la finalidad y el uso de los datos almacenados.

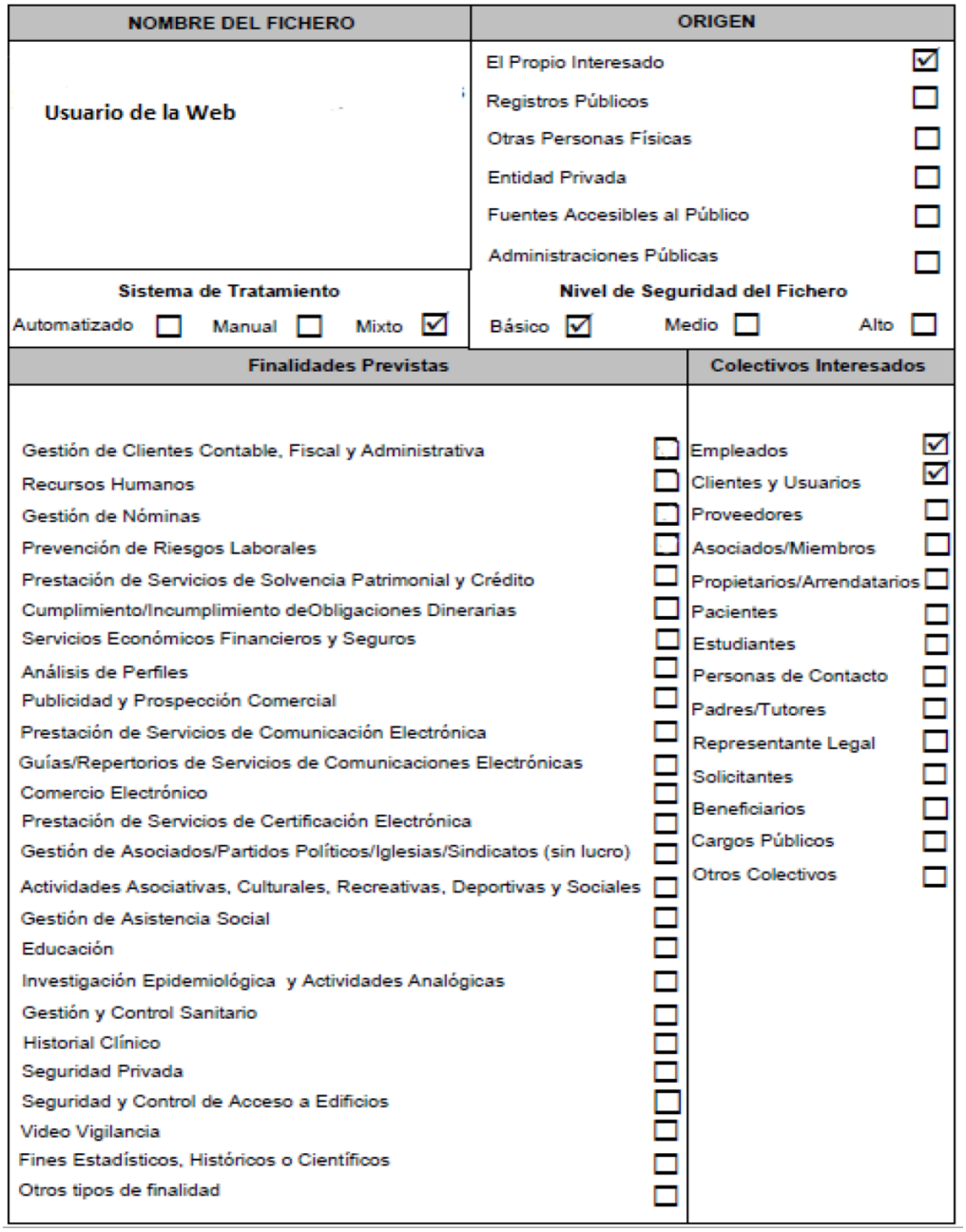
La LOPD distingue entre tres tipos de información.

- **Nivel básico**: datos relacionados con nombres, apellidos, código postal, dirección de domicilio o teléfono.
- **Nivel medio**: datos profesionales (Curriculum Vitae) o datos bancarios.
- **Nivel alto**: Datos relacionados con la salud o información política.

En el caso del trabajo de fin de grado que nos ocupa, la información será toda de nivel básico.

A partir de este punto, la LOPD estipula que el siguiente paso es registrar el fichero con los datos en la LOPD. En este punto, tras el registro del fichero, la LOPD proporcionará un identificador de registro de protección de datos (RGPD) con el que se identifica la información a proteger.

Una vez se tiene dicho identificador es necesario cumplimentar el documento de seguridad con el tipo de datos que se desea proteger. En la Figura 1.1 se puede observar un ejemplo del documento de seguridad. Se puede ver lo preciso del documento en dicha imagen.



NOMBRE DEL FICHERO	ORIGEN
Usuario de la Web	El Propio Interesado ☑ Registros Públicos ☐ Otras Personas Físicas ☐ Entidad Privada ☐ Fuentes Accesibles al Público ☐ Administraciones Públicas ☐
Sistema de Tratamiento Automatizado ☐ Manual ☐ Mixto ☑	**Nivel de Seguridad del Fichero** Básico ☑ Medio ☐ Alto ☐

Finalidades Previstas		Colectivos Interesados	
Gestión de Clientes Contable, Fiscal y Administrativa	☐	Empleados	☑
Recursos Humanos	☐	Clientes y Usuarios	☑
Gestión de Nóminas	☐	Proveedores	☐
Prevención de Riesgos Laborales	☐	Asociados/Miembros	☐
Prestación de Servicios de Solvencia Patrimonial y Crédito	☐	Propietarios/Arrendatarios	☐
Cumplimiento/Incumplimiento deObligaciones Dinerarias	☐	Pacientes	☐
Servicios Económicos Financieros y Seguros	☐	Estudiantes	☐
Análisis de Perfiles	☐	Personas de Contacto	☐
Publicidad y Prospección Comercial	☐	Padres/Tutores	☐
Prestación de Servicios de Comunicación Electrónica	☐	Representante Legal	☐
Guías/Repertorios de Servicios de Comunicaciones Electrónicas	☐	Solicitantes	☐
Comercio Electrónico	☐	Beneficiarios	☐
Prestación de Servicios de Certificación Electrónica	☐	Cargos Públicos	☐
Gestión de Asociados/Partidos Políticos/Iglesias/Sindicatos (sin lucro)	☐	Otros Colectivos	☐
Actividades Asociativas, Culturales, Recreativas, Deportivas y Sociales	☐		
Gestión de Asistencia Social	☐		
Educación	☐		
Investigación Epidemiológica y Actividades Analógicas	☐		
Gestión y Control Sanitario	☐		
Historial Clínico	☐		
Seguridad Privada	☐		
Seguridad y Control de Acceso a Edificios	☐		
Video Vigilancia	☐		
Fines Estadísticos, Históricos o Científicos	☐		
Otros tipos de finalidad	☐		

Figura 1.1: Documento de seguridad para ficheros de la LOPD.

1.4. Estructura de la memoria.

En este apartado de la introducción se explica la estructura de toda la memoria con el fin de poder entender de forma clara el desarrollo que se ha llevado a cabo y que se explica en este documento. La memoria consta de las siguientes secciones:

1- **Introducción**: en esta sección, ya explicada, se ha procedido a mostrar de forma general los objetivos del trabajo además de comentar el marco regulatorio que le afecta.

2- **Estado del Arte**: en esta sección se analizarán las distintas tecnologías existentes para la implementación del sistema y se explicarán los motivos por lo cuales se elige cada una de esas tecnologías por encima de las demás.

3- **Diseño e implementación**: en este apartado en primer lugar se explica el proceso de instalación del entorno elegido en el apartado anterior. Después se exponen los requisitos de la aplicación, se explica la estructura de la base de datos y, por ultimo, todas las funcionalidades de la aplicación para cumplir con los requisitos previamente solicitados.

4- **Validación**: en esta sección se explicarán las pruebas hechas sobre el sistema para validar y comprobar cada uno de los requisitos especificados en la sección anterior

5- **Conclusiones y trabajo futuro**: en esta sección se sacan las conclusiones una vez desarrollado y validado todo el proyecto y, además, se proponen varias ideas para la evolución futura de la aplicación.

6- **Planificación y presupuesto**: en esta ultima sección se analiza la planificación seguida así como el presupuesto necesario para realizar la aplicación.

Capitulo 2: Estado del arte.

Con intención de hacer un desarrollo óptimo del proyecto planteado en la sección anterior, se ha realizado un estudio exhaustivo teórico de las tecnologías en las que se basará el proyecto. [2]

Para poder realizar esta tarea previa al desarrollo de la aplicación se decidió usar un modelo de cliente-servidor.

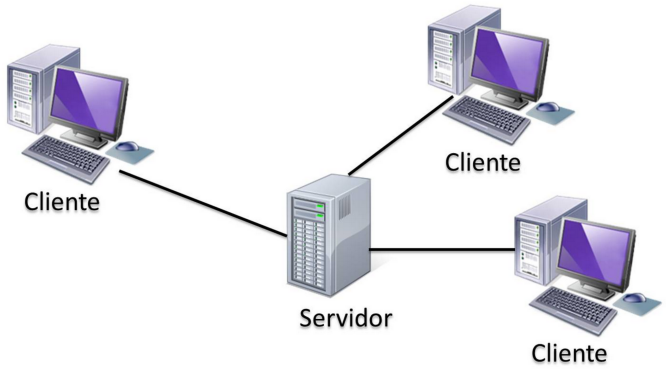


Figura 2.1: arquitectura cliente-servidor.

En la figura 2.1 podemos observar la estructura la estructura cliente-servidor. Se trata de un modelo de aplicación distribuida en el que el sistema se divide en distintos componentes los cuales se ejecutan en entornos diferentes que están conectados a través de una red.

En el caso de la aplicación desarrollada y descrita en este documento, los clientes serán los alumnos y los administradores mientras que el servidor será el sistema que proveerá las funciones de la competición y el encargado de gestionar las interacciones con la base de datos.

2.1. Cliente.

Se ha decidido que el cliente usará un HTML ya que lo que se busca de la aplicación es el aprendizaje por parte de los alumnos en un entorno de competición. Es por esto que el esfuerzo se ha centrado en que la aplicación sea eficiente y con todas las funcionalidades de gestión y de competición necesarias.

2.2. Servidor.

Un servidor es un programa en ejecución que recibe las peticiones de los clientes, las procesa de acuerdo a la lógica del sistema, realiza las operaciones correspondientes de acuerdo al diseño y devuelve una respuesta a los clientes si procede.

Existen multitud de combinaciones tecnológicas para implementar la arquitectura cliente servidor y cada una de estas soluciones está capacitada para comunicarse con todos los elementos necesarios para el sistema para el cual se fueron desarrolladas.

La aplicación que se quiere desarrollar en este proyecto es una aplicación web que además tendrá que gestionar la base de datos necesaria para su funcionamiento. El servidor se dividirá entonces en dos partes: el servidor de aplicaciones web y el sistema de gestión de base de datos.

2.2.1. Servidor de aplicaciones web.

El servidor es un programa que espera peticiones de conexión en un puerto. Dichas peticiones, así como todas las intercambiadas por el cliente y el servidor siguen el protocolo HTTP: “Hypertext Transfer protocol”.

Existen multitud de tipos de servidores de acuerdo al tipo de servicio que prestan o su propósito: de archivos, de correo, proxy, de seguridad, de reserva, web, etc..

Dado que la aplicación que nos concierne es una aplicación web, analizaremos las tres opciones de servidores web que se han considerado más importantes: Internet Information Services (IIS), Nginx y Apache.

2.2.1.1 Internet information services.

Se trata de la solución de Windows para los servidores web que además ofrece servicios de "file transfer protocol” (FTP) [3]. Las principales características de este sistema son:

- Permite la agregación de módulos según los intereses del desarrollador.

- Confiable y seguro.
- No es multiplataforma: solo para Windows.
- Tiene una fácil configuración inicial.
- Soporte nativo para Active Server Pages (ASP) y Microsoft .NET aunque puede incorporar paquetes de otros lenguajes.
- Bajo licencia.

2.2.1.2 Nginx.

Se trata de un servidor web de alto rendimiento que incorpora un servidor proxy para incluir el uso de protocolos de correo electrónico [4]. Las características más llamativas son:

- Software libre y de código abierto.
- Es multiplataforma.
- Tiene balanceo de carga.
- Proporciona soporte de HTTP sobre SSL.
- Bajo consumo de memoria RAM (Random Access Memory) por lo que se considera ligero.

2.2.1.3 Apache.

Desarrollado por la "Apache software foundation" [5], es el servidor web más usado en internet habiendo llegado al 70% del total de los servidores web existentes, aunque en los últimos años su uso ha decrecido hasta alrededor del 60%. Sus principales características son:

- Modular con multitud de paquetes o módulos para distintos propósitos: SSL (secure socket layer), TLS (Transport layer secure), PHP, perl, python, etc.
- Sin animo de lucro.
- Es multiplataforma.
- Es de código abierto, lo que ha favorecido su expansión.
- Apache Tomcat disponible para trabajar con servlets y java server pages (JSP).
- Debido a su alto uso existen multitud de tutoriales y fuentes de información para su desarrollo.
- Muy Robusto.

Debido a que es el web más usado, más estable, multiplataforma y su facilidad de aprendizaje por la gran cantidad de información existente, la opción tecnológica elegida para el servidor web es **Apache**.

2.2.2. Base de datos.

El objetivo principal de los sistemas gestores de base de datos es tratar la información de la manera más sencilla y ordenada posible garantizando así su consistencia además de proveer distintos servicios de seguridad: confidencialidad, integridad y control de acceso. Entre las tecnologías de bases de datos existen distintos tipos según su estructura: jerárquicas, en red, relacionales, no relacionales, etc.

En El caso de la aplicación de este proyecto el análisis se ha centrado en los sistemas gestores de bases de datos relacionales ya que son las más utilizadas por lo que será más sencillo encontrar documentación. También, desde el punto de vista técnico, ni la base de datos ni el número de consultas serán demasiado grandes, por lo que compensa el uso de bases de datos relacionales y no de otros tipos que destacan en estos aspectos.

Se han analizado como posibles alternativas: PostgreSQL, Oracle, Microsoft SQL server y MySQL.

2.2.2.1 PostgreSQL.

Se trata de un sistema de base de datos relacional orientado a objetos. Sus principales características son:

- Alta concurrencia.
- Dispone como tipo de datos direcciones IP, direcciones MAC, figuras geométricas, y arrays.
- Soporta la creación de tipos de datos propios.
- Permite el uso de disparadores o “triggers”, acciones automáticas sobre una tabla bajo determinadas circunstancias.
- Incluye índices, reglas y vistas.
- Permite la inclusión de tipos de usuarios y permisos para cada tipo.
- Muy estable.
- Es multiplataforma.

- Soporta gran cantidad de lenguajes: C, C++, JAVA, PHP, Python, Ruby, TCL, etc.
- Integridad transaccional: si se produce un error la base de datos conserva la consistencia.
- Herramienta de gestión gráfica.
- Gratuita.

2.2.2.2 Oracle Database

Se trata de un sistema de base de datos basado en objetos. [7]

- Posee una alta estabilidad y rendimiento.
- Es multiplataforma.
- Buen soporte para transacciones.
- Buena escalabilidad.
- Precio elevado.

2.2.2.3 Microsoft SQL server

Desarrollado por Microsoft, es la propuesta de esta empresa para la gestión y el modelado de bases de datos. [8]

- Implementa procedimientos almacenados: programas ejecutados en la propia base de datos como respuesta a una petición.
- Permite la declaración de funciones propias.
- Permite la administración de otros servidores.
- Incluye entorno gráfico.
- Consumo de recursos elevado.
- Permite la administración de seguridad a través de permisos.
- Capaz de manejar un volumen alto de datos ya que la cancelación y aceptación de transacciones es muy eficiente.
- Buena seguridad, escalabilidad y estabilidad.
- No es multiplataforma.

2.2.2.4 MySQL.

Este sistema de gestión de base de datos relacional es, además, multihilo y multiusuario, es decir, funciona de manera concurrente y gracias a esto permite la provisión de servicio a más de un usuario a la vez. [9]

- El sistema multihilo basado en los hilos del kernel optimiza el aprovechamiento de los recursos.
- Existen gran cantidad de librerías y herramientas para distintos lenguajes de programación: C, C++, PHP, Java, etc.
- Fácil de usar.
- Configuración de seguridad flexible y buena protección de datos.
- Integridad transaccional.
- Multiplataforma.
- Existencia de versión gratuita así como de versión comercial con licencia.
- Permite Eventos.
- Gran cantidad de tipos de datos definidos.

De entre todos los sistemas de gestión de bases de datos se descartan OracleDB y MicrosoftSQL Server por ser sistemas con coste económico. Microsoft SQL Server también tiene el problema de ser un sistema solo para Windows.

Haciendo una ultima comparativa entre estos dos sistemas:

+ MySQL posee mayor velocidad de operación.
+ Las herramientas de administración de MySQL son más intuitivas y fácilmente configurables.
+ MySQL tiene un bajo consumo de recursos.
+ Uno de los mayores conjuntos para aplicaciones web es el formado por Apache-MySQL-PHP.
- No soporta subconsultas ni disparadores.
- Peor escalabilidad.

+ PostgreSQL tiene mejor escalabilidad.
+ Permite el uso de subconsultas, transacciones y disparadores.
+ Permite el almacenamiento de procedimientos en la base de datos.
- Gran consumo de recursos.
- Más lento que MySQL.

Ya que en la aplicación que nos ocupa se busca una buena velocidad de operación y un bajo consumo de recursos para poder exportar la aplicación a entornos móviles (Android), se elige como tecnología para el desarrollo **MySQL.**

2.2.3. Tecnologías del lado de servidor.

2.2.3.1 Active server pages (ASP).

Es la tecnología de servidor de Microsoft para paginas web generadas dinámicamente [10]. Se comercializa junto con IIS. Las principales características son:

- No es multiplataforma.
- Orientado a objetos.
- Facilidad de mantenimiento de aplicaciones grandes.
- Buena seguridad.
- Alto consumo de recursos.

2.2.3.2 Python

Se trata de un lenguaje de programación de alto nivel que se basa en una sintaxis clara para la fácil lectura del código [11]. Sus características más notables son:

- Se trata de un lenguaje dinámico y versátil, orientado a objetos que permite también programación imperativa (como C o PHP).
- Licencia de código abierto.
- Facilidad de desarrollo gracias a su legibilidad
- Multiplataforma
- Menor documentación ya que su uso esta menos generalizado.
- Sintaxis basada en indentaciones que pueden dificultar su aprendizaje.

2.2.3.3 Java Server Pages (JSP) y Servlets.

Está orientado al desarrollo de paginas web en java. [12]

- Multiplataforma
- Puede utilizar XML y HTML.
- Separa el contenido estático del dinámico.
- Se puede incluir código java.
- Puede ser incrustado en código HTML.
- Aprendizaje complejo.

2.2.3.4 PHP.

Siglas para Hipertext Preprocessor es un lenguaje que puede ser incrustado en HTML. [13]

- Gratuito.
- Multiplataforma.
- Rápido
- Gran cantidad de documentación por su uso extendido y pagina oficial con ejemplos de cada una de sus funciones.
- El cliente solo recibe una pagina HTML por lo que no da problemas de compatibilidad.
- Muy estable
- Fácil aprendizaje y gran cantidad de librerías.
- Compatible con todos los sistemas gestores de bases de datos importantes: MySQL, PostgreSQL, Microsoft SQL server, Oracle, etc.
- El código puede ser de difícil comprensión al estar mezclado con HTML.
- La orientación a objetos no está muy desarrollada.
- Todo el procesado se realiza en el servidor.

De entre todas las opciones tecnológicas para tecnologías del lado de servidor la versatilidad de **PHP**, su fácil aprendizaje y su velocidad hacen que sea la mejor elección.

2.2.4. Conclusión.

Como se ha explicado en esta sección, tras haber realizado el estudio teórico de las principales tecnologías y teniendo en cuenta las siguientes necesidades de la aplicación:

- Velocidad y bajo consumo de recursos: puesto que en el futuro debería poder mudarse el modelo para crear una aplicación móvil y sistemas de duelos de preguntas en tiempo real.
- Facilidad de aprendizaje.
- Estabilidad y multiplataforma.
- Velocidad y consumo de recursos

Teniendo estos objetivos en cuenta y como se ha dicho en los puntos anteriores, **LAMP** es el modelo sistema elegido: Linux, Apache, MySQL, PHP.

2.3. Conclusiones del estado del arte.

Después del análisis teórico realizado se puede concluir que para la aplicación que se quiere implementar la combinación elegida es la óptima y se procederá a continuación a la explicación del diseño y la implementación del sistema siguiendo estas tecnologías.

Capitulo 3: Diseño e implementación.

En esta sección de la memoria se procede a explicar la instalación del entorno, se muestra la tabla de requisitos y se analiza en detalle el diseño de la base de datos y la implementación de la aplicación.

3.1. Instalación y configuración de LAMP.

Una vez decidido este conjunto de tecnologías para el desarrollo del proyecto, se procede a la instalación y configuración del entorno [14], [15].

- Instalación de Linux: en primer lugar se descargó el sistema operativo Ubuntu de su pagina web y se instaló. La versión instalada fue la 14.04 LTS.
- Instalación de MySQL: en la terminal se instaló tanto el cliente como el servidor MySQL:
  apt-get install mysql-server mysql-client
- Instalación de Apache:
  apt-get install apache2
- Instalación PHP:
  apt-get install php5 libapache2-mod-php5

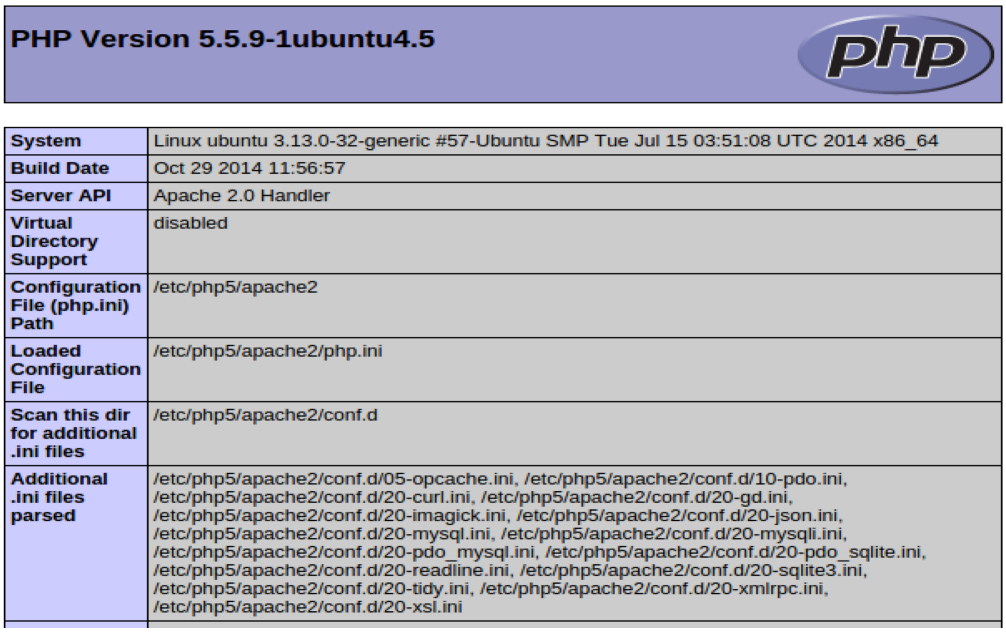
Además de esta instalación básica se tuvo que realizar la instalación del módulo php-mysql.

apt-get install php5-mysql

Y, por ultimo, se reinició el servidor:

/etc/init.d/apache2 restart.

Se comprobó la instalación correcta del sistema mediante la creación de un archivo PHP para, con la función **info()**, sacar la versión de PHP y las extensiones instaladas (ver figura 3.1).



PHP Version 5.5.9-1ubuntu4.5

System	Linux ubuntu 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64
Build Date	Oct 29 2014 11:56:57
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/05-opcache.ini, /etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-curl.ini, /etc/php5/apache2/conf.d/20-gd.ini, /etc/php5/apache2/conf.d/20-imagick.ini, /etc/php5/apache2/conf.d/20-json.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini, /etc/php5/apache2/conf.d/20-pdo_sqlite.ini, /etc/php5/apache2/conf.d/20-readline.ini, /etc/php5/apache2/conf.d/20-sqlite3.ini, /etc/php5/apache2/conf.d/20-tidy.ini, /etc/php5/apache2/conf.d/20-xmlrpc.ini, /etc/php5/apache2/conf.d/20-xsl.ini

Figura 3.1: versión y extensiones instaladas.

3.2. Requisitos.

Una vez instalado el sistema se hizo una lista de requisitos que debía cumplimentar el proyecto con el fin de poder crear la aplicacion y que esta trabajase de la manera deseada. Se resumen en la siguiente tabla.

# Requisito	Requisito
	Requisitos generales:
Requisito # 1	Deben existir dos tipos de usuarios: 1- Alumno. 2- Administrador.
Requisito # 2	Cada usuario debe tener todos los datos necesarios para su correcta gestión por parte de la aplicación y del la administración, incluidos los usuarios administradores.
Requisito # 3	La información de todos los usuarios debe estar guardada en una tabla independiente en la base de datos.
Requisito # 4	Las contraseñas deben estar cifradas.
Requisito # 5	La aplicación debe utilizar sesiones de usuario.
	Los usuarios con permisos de administrador deben poder realizar las siguientes operaciones:
Requisito # 6	Registrar usuarios.

Requisito # 7	Registrar usuario.
Requisito # 8	Modificar usuario.
Requisito # 9	Mostrar usuarios.
Requisito # 10	Borrar usuario.
Requisito # 11	Subir preguntas.
Requisito # 12	Modificar pregunta
Requisito # 13	Mostrar preguntas.
Requisito # 14	Borrar pregunta.
Requisito # 15	Crear competición.
Requisito # 16	Modificar competición.
Requisito # 17	Mostrar competiciones.
Requisito # 18	Borrar competición.
Requisito # 19	Clasificación.
	Los usuarios con permisos de alumno deben poder realizar las siguientes operaciones:
Requisito # 20	Cambiar contraseña.
Requisito # 21	Realizar competición.
Requisito # 22	Clasificación.
	Requisitos de operaciones:
Requisito # 23	La operación “Registrar usuarios” debe poder registrar más de un usuario desde un archivo CSV.
Requisito # 24	La operación “Registrar usuario” debe registrar usuarios de uno en uno.
Requisito # 25	La operación “Modificar usuario” debe mostrar los datos actuales del usuario y mostrar un formulario para su modificación.
Requisito # 26	La operación “mostrar datos de los usuarios” debe mostrar la totalidad de la tabla de usuarios.
Requisito # 27	La operación “borrar usuario” debe eliminar el usuario de la base de datos.

Requisito # 28	La información de todas las preguntas debe estar guardada en una tabla independiente en la base de datos.
Requisito # 29	La operación “subir preguntas” debe subir las preguntas desde un archivo CSV.
Requisito # 30	La operación modificar pregunta debe mostrar los datos de la pregunta escogida y un formulario para su modificación.
Requisito # 31	La operación “mostrar preguntas” debe mostrar la totalidad de preguntas después de haber elegido un tema del que mostrar las preguntas.
Requisito # 32	La operación “borrar pregunta” debe borrar la pregunta seleccionada.
Requisito # 33	La información de todas las competiciones debe estar guardada en una tabla independiente en la base de datos.
Requisito # 34	La operación “crear competición” debe crear una competición de acuerdo a los parámetros solicitados en el formulario.
Requisito # 35	La operación “modificar competición” debe mostrar los datos de la competición y un formulario para su modificación.
Requisito # 36	La operación “mostrar datos de las competiciones” debe mostrar los datos de todas las competiciones existentes.
Requisito # 37	La operación “mostrar clasificación” debe mostrar la clasificación de la competición seleccionada.
	Requisitos de usuarios:
Requisito # 38	La operación “cambiar contraseña” debe solicitar una nueva contraseña al usuario y cambiarla por la vieja en la tabla de usuarios.
Requisito # 39	La operación “mostrar competiciones” debe mostrar por pantalla al usuario las competiciones activas en ese momento y dar la posibilidad de elegir cualquiera de ellas para su realización.
Requisito # 40	La operación “mostrar clasificación” debe mostrar la clasificación del usuario de esa sesión tanto para el grupo magistral como para el grupo reducido.

Figura 3.2: tabla de requisitos.

3.3. Diseño e implementación.

Se decidió llevar a cabo este proyecto de aplicación web usando el patrón modelo-vista-controlador para luego poder exportarlo de manera sencilla a otras plataformas en la continuación de este proyecto por parte de otro alumno. [16]

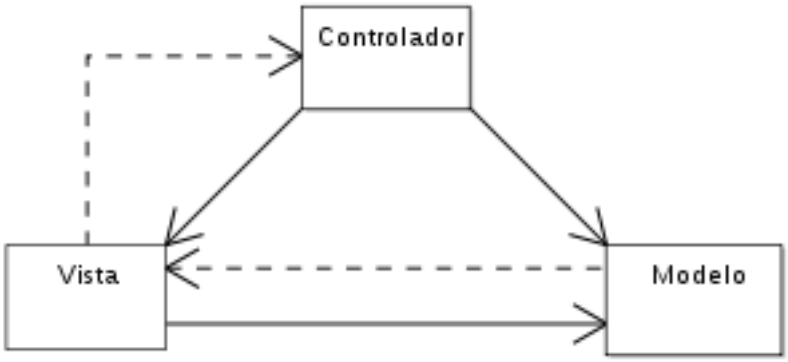


Figura 3.3: esquema del patrón modelo-vista-controlador.

Este es un modelo o patrón divide la arquitectura en 3 componentes:

1- **El modelo**: es el que controla la lógica de negocio de la aplicación, es decir, el cerebro del programa. Envía a la vista la información que debe mostrar y recibe las peticiones del controlador.

2- **El controlador**: Responde a las peticiones de los usuarios enviando eventos al modelo.

3- **La vista**: presenta el modelo, es decir, muestra la información que el sistema quiere mostrarle al usuario en cada momento.

3.3.1. Modelo de datos.

Esta parte del documento explica el diseño de la base de datos de la aplicación: se describe la estructura organizativa de los datos en las distintas tablas además de cómo están almacenados y qué operaciones se pueden hacer sobre ellos.

Teniendo una idea general del sistema tras haber especificado la tabla de los requisitos, la aplicación tendrá una base de datos que constará de 3 tablas fijas y un número variable de tablas tantas como competiciones se generen. Este funcionamiento se explicará más adelante.

3.3.1.1 Tabla de usuarios: "users".

La tabla de usuarios es única para todos los usuarios participantes. Es la tabla donde se guarda toda la información necesaria para la gestión de usuarios, la gestión de los grupos a los que pertenecen y los permisos de cada uno de ellos. Es también donde se guardan los datos del administrador.

Cada usuario corresponderá a una fila de la tabla y ésta consta de las siguientes columnas:

- Nombre.
- Primer apellido.
- Segundo apellido.
- ID: Se utilizará el número de identificación de alumno.
- Correo electrónico.
- Contraseña: será proporcionada por el administrador.
- Grupo magistral.
- Grupo reducido.
- Permisos.

Las operaciones del menú de administrador que tendrán relación con la tabla de usuarios son las de "Registrar usuarios", "registrar usuario", "borrar usuario" y "modificar usuario".

Por su parte, el alumno solo podrá modificar su contraseña con la operación "modificar contraseña". Adicionalmente, se incluye la operación de "mostrar datos de los usuarios" que accederá a la tabla a modo de lectura.

La creación de esta tabla así como su inicialización se realizan de manera previa a la puesta en marcha de la aplicación, puesto que, si no, no seria posible para el administrador acceder a su menú de aplicación y no podría empezar a preparar el sistema para su puesta en marcha.

3.3.1.2 Tabla de Preguntas: "questions".

La tabla de preguntas de la aplicación también es única para todas las preguntas que queramos tener en el sistema. De esta tabla se podrán sacar las preguntas para las competiciones cuando sean creadas.

Cada fila corresponderá a una pregunta y tendrá las siguientes columnas:

- Identificador de pregunta.
- Tema de la asignatura al que pertenecen.
- Enunciado.
- 4 posibles respuestas.
- Respuesta correcta.
- Nivel.

En el caso de la tabla de preguntas solo el administrador tendrá acceso a modificar su contenido. Esto será posible a través de las operaciones "subir preguntas", "modificar pregunta" y "borrar pregunta". Adicionalmente, como en el caso anterior, se incluye la operación "mostrar datos de las preguntas" como función de lectura.

El parámetro "id" de la tabla, es decir, el identificador de cada pregunta, es único y se genera automáticamente de acuerdo a cada pregunta que se inserta.

3.3.1.3 Tabla de competiciones: "competitions".

Por ultimo, la tercera de las tablas fijas es la tabla de competiciones. En esta tabla se guardará toda la información necesaria para la gestión de las competiciones y los parámetros con los que se creó dicha competición.

Cada competición se corresponderá con cada fila de la tabla y ésta tendrá las siguientes columnas:

- Nombre de la competición.
- Temas incluidos.
- Grupos.
- Número de preguntas.
- Fecha de inicio.
- Fecha de fin.

Las operaciones con las que el administrador puede interaccionar con esta tabla son "crear competición" y "modificar competición". Se incluye también la operación "Mostrar datos de las competiciones" de carácter consultivo.

Los únicos parámetros que serán variables serán las fechas de inicio y fin de la competición, puesto que, como se explicará más adelante, el resto de parámetros de la competición afectan a la creación de la tabla de preguntas.

Además de las 3 tablas fijas, cada operación "crear competición" creará dos tablas adicionales asociadas a la competición: la tabla de preguntas de la competición y la tabla de clasificación de la competición.

3.3.1.4 Tabla de preguntas de la competición: "[nombre de la competicion]_questions_table".

Esta tabla tendrá como nombre el nombre de la competición seguido de la cadena "_questions_table" y, en ella, cada fila corresponderá a un usuario participante en la competición y cada columna a partir de la segunda corresponderá a una pregunta distinta.

3.3.1.5 Tabla de clasificación de la competición: "[nombre de la competicion]_classification_table".

Esta tabla tendrá como nombre el nombre de la competición seguido de la cadena "_classification_table" y, en ella, cada fila corresponderá a un usuario participante en la competición y su segunda columna será la suma de puntos obtenidos en la competición.

El diagrama relación de las tablas de la base de datos se muestra en la figura 3.4.

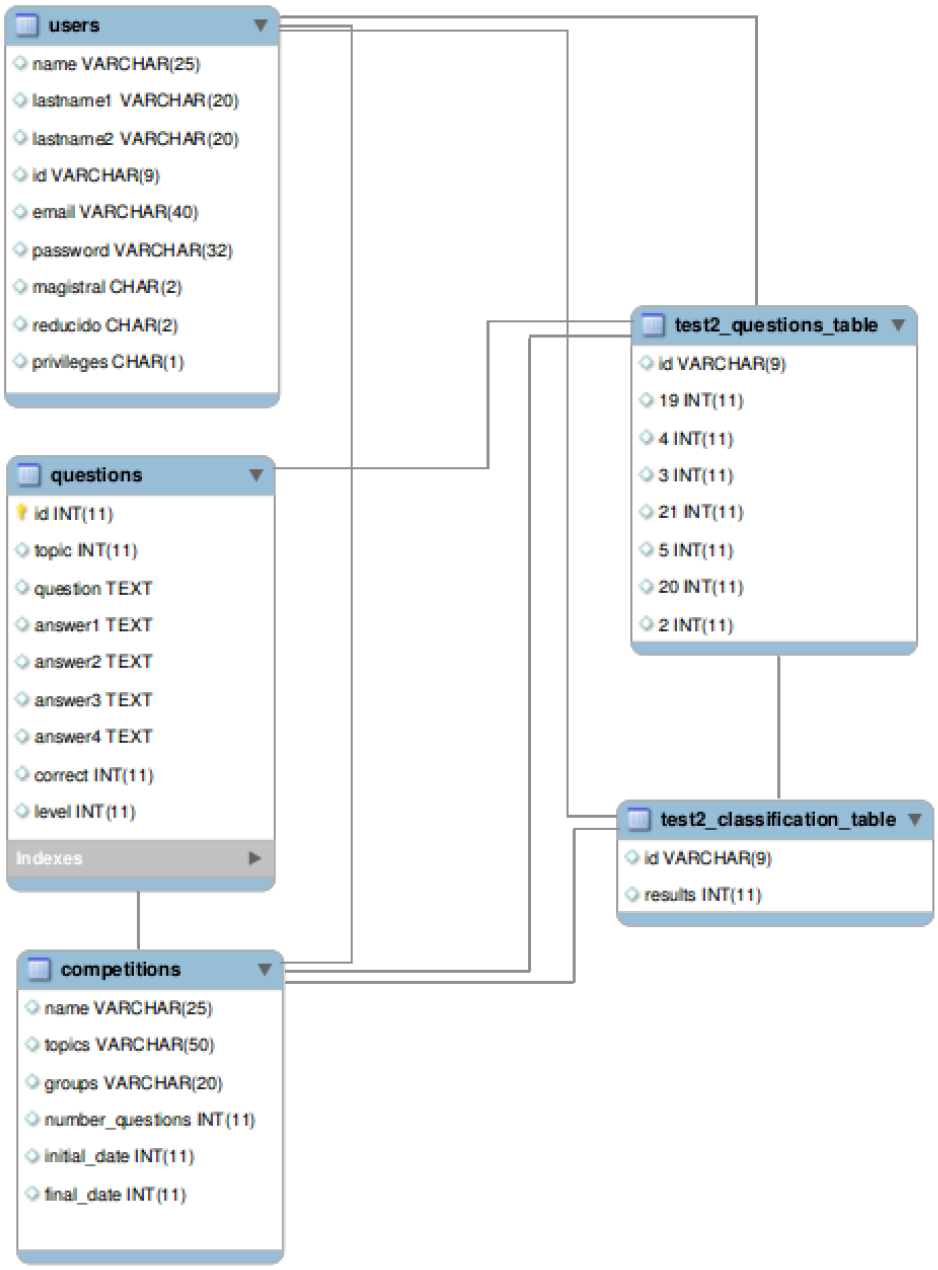


Figura 3.4: diagrama relacional de la base de datos diseñada para la aplicación.

3.3.2. Funciones.

A lo largo del desarrollo del proyecto se han ido implementando una serie de funciones en PHP para manipular las distintas tablas en la base de datos. Todas estas funciones se emplean una o más veces a lo largo de la implementación de la aplicación. Estas funciones se encuentran en el fichero database_functions.php y se exponen continuación se exponen de manera detallada. [17][18].

# Función	Nombre Función	Explicación Función	Operación SQL
# 1	**Connect ()**	Conecta el servidor con la base de datos.	
# 2	**check_login (email, password)**	Con la propia consulta comprueba que exista el usuario en la base de datos y que la contraseña sea correcta.	Select
# 3	**insert_user (data)**	Inserta los datos del usuario nuevo al final de la tabla de usuarios.	Insert
# 4	**get_user (id)**	Devuelve todos los datos del usuario especificado por el parámetro id si existiera.	Select
# 5	**show_users ()**	Devuelve todos los datos de todos los usuarios de la tabla.	Select
# 6	**show_question (id)**	Devuelve la pregunta indicada por el parámetro id si la hubiera.	Select
# 7	**insert_question (data)**	Inserta los datos de la nueva pregunta al final de la tabla de preguntas.	Insert
# 8	**modify_user (data, id)**	Actualiza la información del usuario id con los datos que se le pasan como parámetro.	Update
# 9	**modify_question (data, id)**	Actualiza la información de la pregunta id con los datos que se le pasan como parámetro.	Update
# 10	**delete_user (id)**	Borra de la tabla de usuarios la fila correspondiente al usuario que se le pasa como parámetro.	Delete
# 11	**delete_question (id)**	Borra de la tabla de preguntas la fila correspondiente al id de pregunta que se le pasa como parámetro.	Delete

# 12	**show_questions_topics ()**	Devuelve los temas que tienen preguntas disponibles en la tabla de preguntas. Por ejemplo: temas 1, 2 y 4.	Select
# 13	**show_users_groups ()**	Devuelve los grupos magistrales y reducidos existentes en la tabla de usuarios. Por ejemplo: 91, 92 y 95.	Select
# 14	**get_group_list (group, group_type)**	Devuelve la lista de identificadores de usuario del grupo especificado. Por ejemplo: 91, reducido.	Select
# 15	**get_all_users ()**	Devuelve la lista de identificadores de toda la tabla de usuarios.	Select
# 16	**insert_competition (data)**	Inserta en la tabla de competiciones una nueva fila con los datos de la nueva competición.	Insert
# 17	**add_table_user (table_name, id)**	Inserta el usuario id en la tabla especificada por table_name	Insert
# 18	**create_competition_tables (data)**	Primero llama a la función insert_competition () y después crea la tabla de preguntas y la tabla de clasificación para la nueva competición de acuerdo a los datos de entrada. Esta función es dinámica y tendrá especial análisis en el apartado de funcionalidades.	Create
# 19	**add_table_question (table_name, id)**	Añade a la tabla especificada una columna con el id de pregunta indicado.	Alter
# 20	**get_question (id)**	Devuelve toda la información de la pregunta especificada por id.	Select
# 21	**get_group_for_**	Devuelve los grupos magistral y	Select

	user (id)	reducido a los que pertenece el usuario indicado por id.	
# 22	**get_competitions_reducido (group)**	Devuelve los nombres de las competiciones que tiene el grupo reducido indicado.	Select
# 23	**get_competitions_magistral (group)**	Devuelve los nombres de las competiciones que tiene el grupo magistral indicado.	Select
# 24	**get_competition_times (name)**	Devuelve la fecha y hora inicial y final de la competición indicada.	Select
# 25	**get_competition_questions (name)**	Devuelve una lista de todas las preguntas de la competición indicada.	Select
# 26	**show_topic_questions (topic)**	Devuelve una lista de todas las preguntas del tema indicado.	Select
# 27	**show_competitions ()**	Devuelve una lista de todas las competiciones ya existentes.	Select
# 28	**get_random_questions (topics)**	Devuelve una lista ordenada aleatoriamente con todos los identificadores de pregunta de la tabla de preguntas que pertenezcan a los temas indicados por el parámetro.	Select
# 29	**create_questions_table (name)**	Crea la tabla de preguntas de competición con el nombre especificado.	Create
# 30	**create_clasification_table (name)**	Crea la tabla de clasificación de competición con el nombre especificado.	Create
# 31	**modify_competition_dates(competition_name, data)**	Modifica las fechas de inicio y final de la tabla indicada por el parámetro.	Update

# 32	**delete competition (competition_name)**	Borra la competición indicada por el parámetro.	Delete
# 33	**delete_competition_questions_table (name)**	Borra la tabla de preguntas de competición indicada por el parámetro.	Delete
# 34	**delete_competition_classification_table (name)**	Borra la tabla de clasificación de competición indicada por el parámetro.	Delete
# 35	**show_classification(competition_name)**	Devuelve la tabla de clasificación de la competición indicada por el parámetro.	Select
# 36	**insert_result (competition_name, id, question_id, result)**	Modifica en la tabla de preguntas de la competición indicada la fila del usuario en columna de la pregunta el valor del resultado de la pregunta.	Update
#37	**check_actual_question(competition_name, user_id)**	Esta función devuelve el número de pregunta que le corresponde al alumno en dicha competición.	Select

Figura 3.5: tabla de funciones de acceso a la base de datos.

3.3.3. Funcionalidades.

Control de acceso.

Como se ha podido comprobar en la lista de requisitos existen dos tipos de usuarios: los administradores y los alumnos. Cada uno de estos dos tipos de usuarios tienen un valor de permisos asociado. En el caso de la aplicación, esta variable tiene dos valores posibles: 1 y 0. En el caso de los usuarios administradores tendrá el valor 1, lo que les dará acceso al menú de administrador y a todas las operaciones de administrador. En el caso de los

usuarios alumnos el valor de la variable es 0, lo que les proporcionará acceso a las operaciones de alumnos. Todas estas operaciones se explicarán a lo largo de esta sección.

Teniendo esto en cuenta, en todas las páginas web que componen la aplicación se ha incluido una condición que permite o no el acceso al usuario de acuerdo a sus permisos, dividiendo las páginas en dos grupos: "solo para alumno" y "solo para administrador". Algunas de estas páginas son comunes a ambos tipos de usuario o comparten partes de acuerdo al diseño.

Este control se lleva a cabo mediante la variable $_SESSION de PHP que se inicializa a unos parámetros determinados al hacer el Login de usuario. Esto se explicará en el siguiente punto.

Operación de Login.

La primera vista que tenemos de la aplicación web es el formulario de acceso a ella. Al introducir el correo electrónico de un usuario (ya sea administrador o alumno) y la contraseña el usuario hace una petición de validación al servidor para acceder al menú de la aplicación, donde le serán mostradas todas las opciones de las que dispone. El diagrama de flujo del funcionamiento del login se puede ver en la figura 3.6:

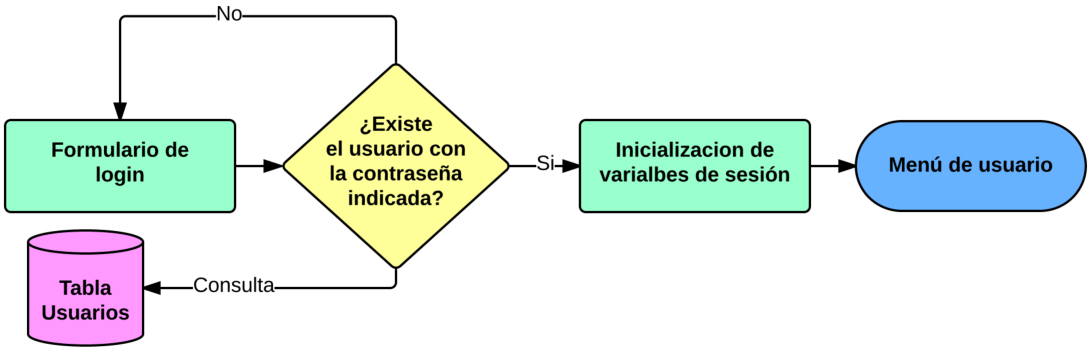


Figura 3.6: diagrama de flujo de la operación Login.

Una vez el servidor ha comprobado con la operación **check_login()** que el usuario existe y su contraseña es la correcta, procede a la inicialización de la variable de sesión $_SESSION con el nombre de usuario y sus permisos asociados.

Esta variable de sesión guardará la información del usuario durante un tiempo de 24 minutos (tiempo por defecto de duración de las variables de sesión en PHP) y se reiniciará cada vez que se actualice el navegador o cambie de pagina la aplicación.

NOTA: Dado que el uso de la aplicación es rápido no se ha considerado necesario ampliar este parámetro de tiempo en ninguna de las paginas que componen la aplicación .

Como se ha explicado previamente en el control de acceso, cada parte de la aplicación comprobará los permisos en la variable de sesión y permitirá la actuación de cada usuario en consecuencia.

En caso de error en la operación de "Login", ya sea porque el usuario o la contraseña son incorrectos o bien porque se ha dejado algún campo del formulario vacío, regresará al formulario y notificará al usuario un mensaje de error.

Operaciones de administración.

Las operaciones de administración se pueden clasificar en cuatro grupos según la parte del sistema sobre la que operan: operaciones sobre usuarios, preguntas, sobre competición y mostrar clasificación. Esto se ve reflejado en la figura 3.7.

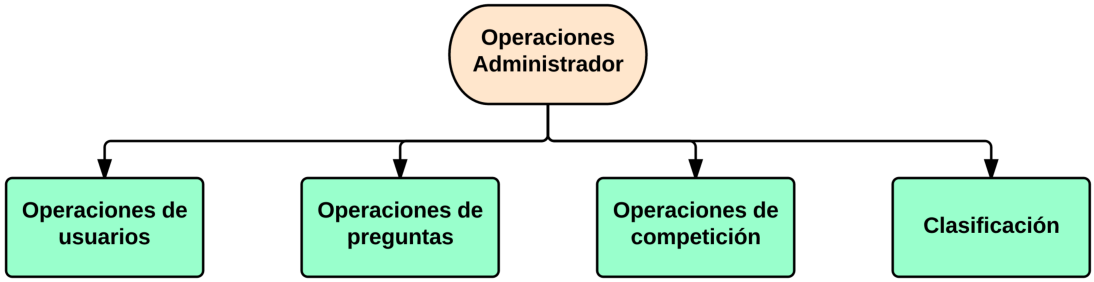


Figura 3.7: Clasificación de las operaciones de administración.

3.3.3.1 Operaciones de usuarios:

Se puede observar en el diagrama 3.8 cada una de las operaciones de usuarios que se mencionaron en los requisitos.

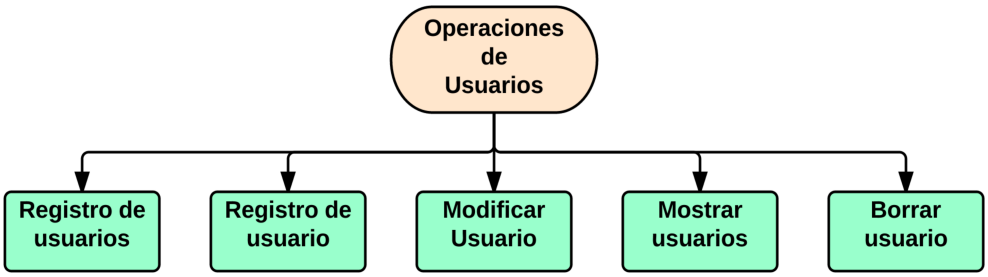


Figura 3.8: operaciones de usuarios.

Como se puede observar, el registro de los usuarios en la aplicación tiene dos modalidades distintas: registro de usuarios y registro de usuario. Se explican todas las operaciones a continuación.

1- **Registro de usuarios**. El diagrama de flujo de esta operación puede verse en la figura 3.9:

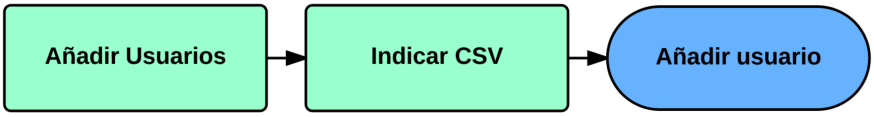


Figura 3.9: diagrama de flujo de la operación "Registro de usuarios".

NOTA: los archivos CSV (del inglés, "comma separated values") son un tipo de archivos que guardan los datos en forma de tabla separados por comas. En este caso, se ha cambiado el tipo de separador al símbolo barra vertical: "|". En la figura 3.10 se puede ver cómo sería la tabla en Excel de 5 usuarios y en la figura 3.11 se ve la vista del archivo una vez guardado utilizando el comando "cat" de Linux.

	A	B	C	D	E	F	G	H	I
1	Ignacio	Ortiz	Luengo	100072644	100072644@alumnos.uc3m.es	password1	95	95	0
2	Usuario1	Apellido1 usuario1	Apellido2 usuario1	100072645	100072645@alumnos.uc3m.es	password2	95	95	0
3	Usuario2	Apellido1 usuario2	Apellido2 usuario2	100072646	100072646@alumnos.uc3m.es	password3	91	91	0
4	Usuario3	Apellido1 usuario3	Apellido2 usuario3	100072647	100072647@alumnos.uc3m.es	password3	95	92	0
5	Usuario4	Apellido1 usuario4	Apellido2 usuario4	100072648	100072648@alumnos.uc3m.es	password4	91	92	0

Figura 3.10: tabla de una lista de usuarios.

```
Ignacio|Ortiz|Luengo|100072644|100072644@alumnos.uc3m.es|password1|95|95|0
Usuario1|Apellido1 usuario1|Apellido2 usuario1|100072645|100072645@alumnos.uc3m.es|password2|95|95|0
Usuario2|Apellido1 usuario2|Apellido2 usuario2|100072646|100072646@alumnos.uc3m.es|password3|91|91|0
Usuario3|Apellido1 usuario3|Apellido2 usuario3|100072647|100072647@alumnos.uc3m.es|password3|95|92|0
Usuario4|Apellido1 usuario4|Apellido2 usuario4|100072648|100072648@alumnos.uc3m.es|password4|91|92|0
```

Figura 3.11: "cat" del archivo CSV generado.

Cuando se llama a la operación insertar usuarios se genera una pagina HTML en la que se pregunta al administrador el archivo CSV desde el que quiere cargar los datos de los usuarios. Una vez se especifica el archivo y se envía la petición al servidor, el servidor llama a la función **get_all_users()**, que devuelve

todos los usuarios registrados en la base de datos en un array, se comprueba cada nuevo usuario que se quiere registrar y, si no esta ya registrado, se inserta en la tabla llamando a la función **insert_user()**. Esta comprobación y, si cumple las condiciones, posterior inserción se repite tantas veces como filas tenga el archivo CSV. Como apoyo se muestra el diagrama de flujo de este proceso en la figura 3.12

Nótese que el cifrado de la contraseña se realiza en el servidor, antes de insertar la contraseña en la tabla junto con el resto de los datos. Se utiliza la función MD5() de PHP en la función **insert_user()**.

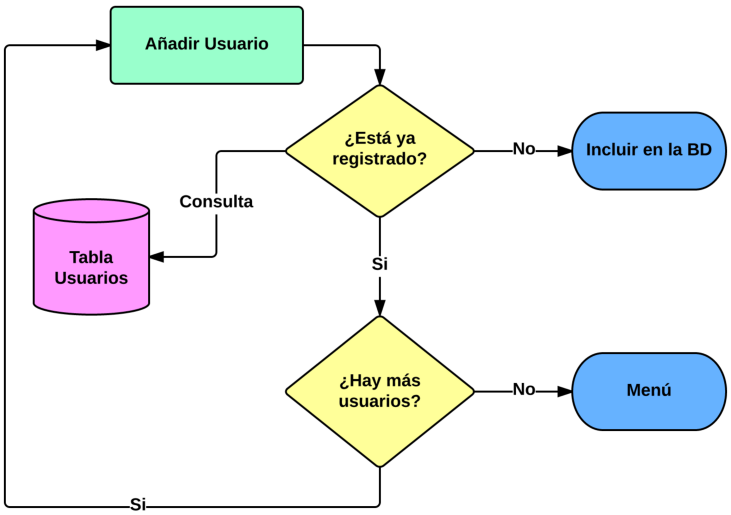


Figura 3.12: Diagrama de flujo interno de la operación "registrar usuarios.

En caso de error en la especificación del archivo CSV, se le notifica al administrador mediante un aviso.

2- **Registro de usuario**: se diferencia de la operación de registro de usuarios en que por esta vía de registro solo se pueden registrar usuarios de uno en uno. Su diagrama de flujo es el siguiente:

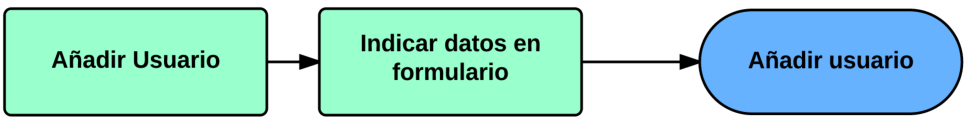


Figura 3.13: diagrama de flujo de la operación "Añadir usuario".

La idea de hacer estas dos modalidades es que el administrador pueda crear una tabla en Excel (por ejemplo) con todos los alumnos de un grupo e

ingresarlos en una sola operación y, si luego cambia la lista de clase o se olvida de algún alumno, pueda ingresar los usuarios nuevos de manera sencilla.

El funcionamiento es parecido al de la función anterior: primero se genera un formulario HTML para que el administrador ingrese los datos del nuevo usuario, después se comprueba que ningún campo en del formulario estuviese vacío para, si esto se cumple, comprobar que el usuario no estuviese ya incluido en la tabla y, en ese caso, introducirlo. Las funciones utilizadas son, de nuevo**, get_all_users()** e **insert_user().**

En caso de error se notifica al administrador, o bien con un mensaje de "formulario incompleto" o bien con un mensaje de "usuario ya registrado".

3- **Modificar usuario**: esta operación primero despliega por pantalla todos los usuarios existentes en la tabla y después pregunta en un formulario HTML el ID del usuario que quiere modificar. Su diagrama de flujo puede verse a continuación:

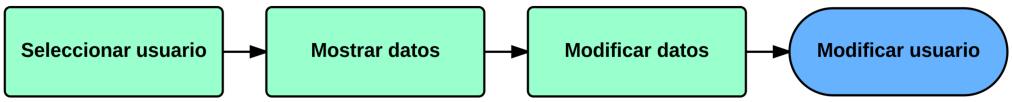


Figura 3.14: diagrama de flujo de la operación "Modificar usuario".

De acuerdo al ID introducido se llama a la función **get_user()**, que devuelve los datos del usuario que se quiere borrar y crea un nuevo formulario dichos datos ya introducidos en el mismo de manera que el administrador solo deberá modificar los campos que quiera cambiar. Tras aceptar, los datos del usuario serán modificados en la tabla de usuarios de la base de datos llamando a la función **modify_user()**.

En caso de que no se especifique nada en el formulario, de que el usuario no exista en la tabla o de que no se haya podido modificar el usuario, el administrador recibirá un mensaje de error.

4- **Mostrar Usuarios:** esta operación es de carácter puramente consultivo a la base de datos, es decir, no altera de ninguna forma la tabla de usuarios. Despliega por pantalla en una tabla HTML los datos de todos los usuarios incluidos en la aplicación.

La operación mostrar usuarios se realiza llamando a una función llamada s**how_users()** que realiza una consulta de todos los datos de los usuarios a la base de datos y crea una tabla HTML con los datos obtenidos.

5- **Borrar Usuario**: esta operación primero despliega por pantalla todos los usuarios existentes en la tabla y después pregunta en un formulario HTML el ID del usuario que quiere borrar. Su diagrama de flujo se muestra a continuación:

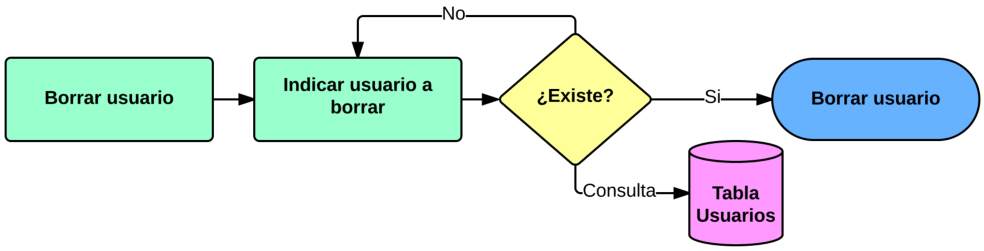


Figura 3.15: diagrama de flujo de la operación "Borrar usuario".

La operación de borrado es llevada a cabo por la función **delete_user().** En caso de que no se especifique un usuario en el formulario, el usuario no exista en la base de datos o no se haya podido borrar, se devuelve un mensaje de error al administrador.

3.3.3.2 Operaciones de preguntas:

Se puede observar en el diagrama 3.16 cada una de las operaciones de preguntas que se mencionaron en los requisitos.

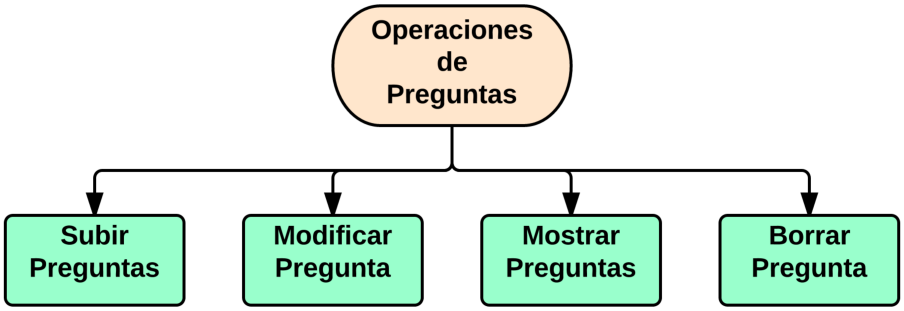


Figura 3.16: Operaciones de preguntas.

Se procede a explicar cada una de las operaciones:

1- **Subir preguntas:** se muestra el diagrama de flujo en la figura 3.17**.**

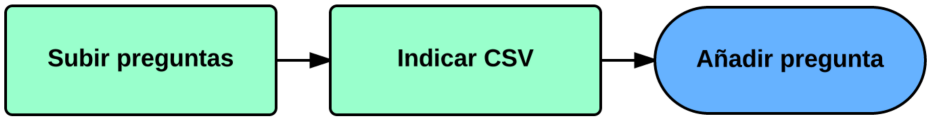


Figura 3.17 diagrama de flujo de la operación "Subir preguntas".

Cuando se llama a la operación subir preguntas se genera una pagina HTML en la que se solicita al administrador que indique el archivo CSV desde el que quiere cargar las preguntas a insertar en la tabla. Una vez se especifica el archivo, este se envía al servidor y se van añadiendo las preguntas una a una. El diagrama de flujo de la figura 3.18 muestra el proceso seguido.

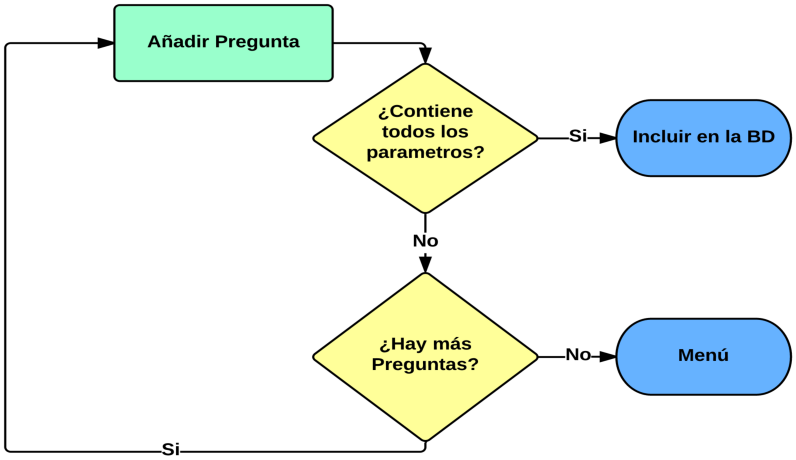


Figura 3.18: Diagrama de flujo interno de la operación "Subir preguntas"

Teniendo ya el archivo CSV, se procede a leer cada fila de la tabla y, si contiene todos los parámetros necesarios para completar toda una fila de la tabla de preguntas, se añade la pregunta a dicha tabla con la operación **insert_question()**. Este proceso se repite tantas veces como filas haya en el CSV.

En caso de que alguna pregunta no contenga todos los campos no se incluye en la tabla. Si no se especifica un archivo CSV se notifica un mensaje de error al administrador.

2- **Modificar pregunta**: el diagrama de flujo de esta operación se muestra en la figura 3.19.

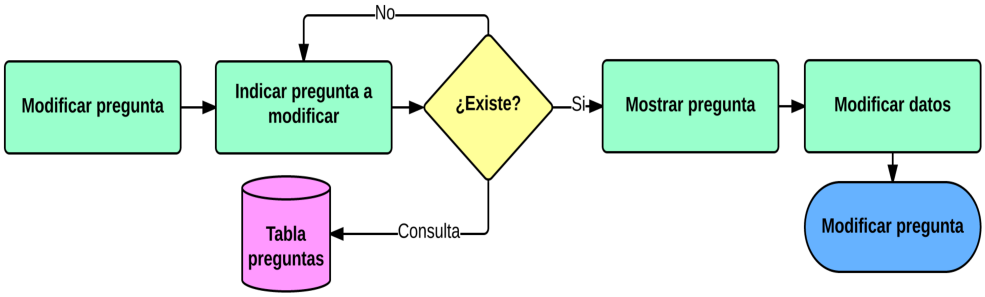


Figura 3.19: diagrama de flujo de la operación "Modificar pregunta".

Esta operación primero pregunta por pantalla la pregunta que se desea modificar. Una vez introducido el parámetro comprueba en la base de datos su existencia y, en ese caso, solicita la información de la pregunta mediante la función **show_question()**. Muestra la pregunta obtenida por pantalla y, dentro de un formulario, se solicitan los datos nuevos en los campos que se quieren modificar. Una vez el administrador ha modificado los datos de la pregunta, la modificación en la base de datos se realiza mediante la función **modify_question()**.

En caso de que no se especifique un id, el id no sea válido o no se haya podido modificar la pregunta se notifica al administrador mediante un mensaje de error.

3- **Mostrar preguntas**: esta operación es, al igual que "Mostrar usuarios" una operación de carácter consultivo a la base de datos. Su diagrama de flujo se muestra en la figura 3.20.

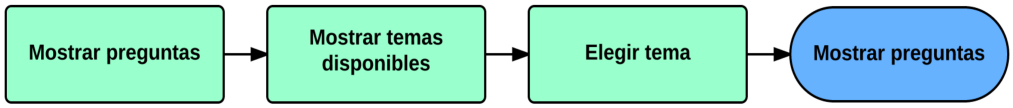


Figura 3.20: diagrama de flujo de la operación "Mostrar preguntas".

Esta operación primero muestra por pantalla al administrador los distintos temas de los que hay disponibles preguntas. Esta consulta se realiza con la operación **show_questions_topics()**. El administrador elige el tema del que quiere ver las preguntas y se le muestran por pantalla todas las preguntas de ese tema con la función **show_topic_questions().**

Se decidió mostrar las preguntas por temas porque en un sistema con un elevado número de preguntas la pagina web que se generaría podría ser muy grande.

En caso de que seleccione un tema que no exista se notifica un mensaje de error.

4- **Borrar pregunta**: el diagrama de flujo de esta operación se muestra en la figura 3.21.

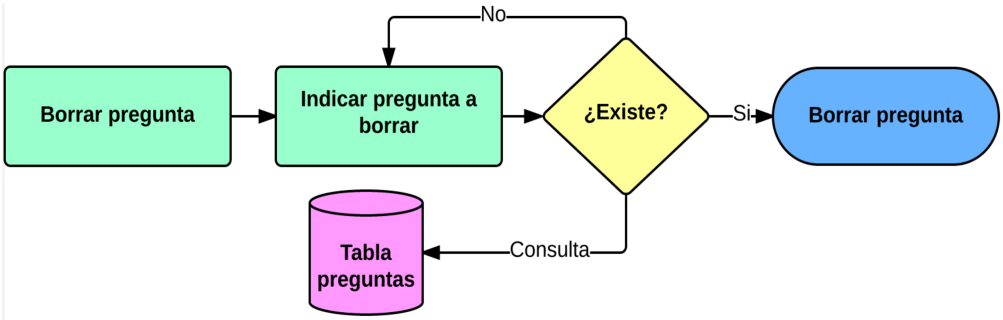


Figura 3.21: diagrama de flujo de la operación "Borrar pregunta".

Esta operación pregunta al administrador mediante un formulario el identificador de pregunta que quiere borrar. Esta información el administrador la ha podido consultar con la operación "Mostrar preguntas" descrita previamente. Si el identificador es correcto, se elimina la pregunta llamando a la función **delete_question()**.

En caso de que no se pueda borrar la pregunta, no se haya introducido el identificador o éste sea inválido se notifica un mensaje de error al administrador.

3.3.3.3 Operaciones de competición:

Se puede observar en el diagrama 3.22 cada una de las operaciones de competición que se mencionaron en los requisitos.

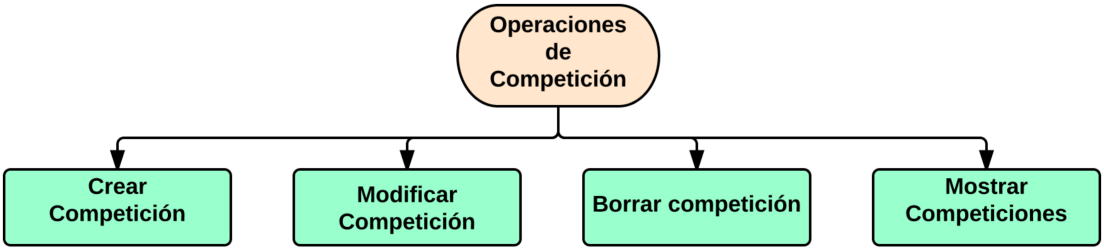


Figura 3.22: operaciones de competición.

1- **Crear competición:** es la operación de creación de todo lo necesario para que un alumno pueda realizar una competición. El diagrama de flujo de la operación se muestra en la figura 3.23.

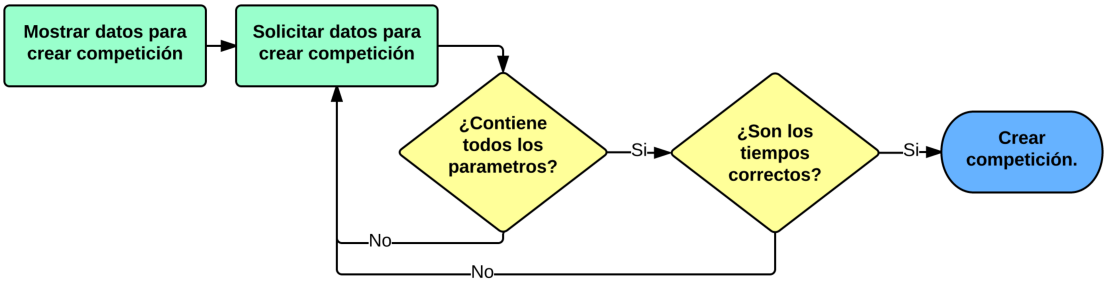


Figura 3.23: diagrama de flujo de la operación "Crear competición".

En primer lugar el sistema muestra información relevante de lo que debe saber el administrador para poder crear la competición correctamente. Esta información es:

- **Temas disponibles**: mediante la operación **show_questions_topics().**

- **Grupos disponibles**: mediante la operación **show_users_groups().**

- **Competiciones ya existentes**: mediante la operación **show _competitions().**

Esta información indica al administrador los posibles valores de los parámetros de diseño de la nueva competición que va a crear.

A continuación, en la misma página, se genera un formulario HTML para que el administrador elija los parámetros de diseño. Estos son:

- **Nombre**: será el nombre de la competición y a partir de el se crearán 2 tablas en la base de datos que, junto con la fila de la competición en la tabla de competiciones servirán para controlar la competición.

- **Temas**: se indicarán, separados por comas, los temas que se quieren incluir en la competición.

- **Grupos**: se indica el número del grupo para el que se quiere crear la competición. Por ejemplo: 95.

- **Número de preguntas**: el número de preguntas que se quieren incluir en la competición.

- **Fecha y hora de inicio**: con este parámetro el administrador indica la fecha y hora a la que quiere que se active la competición para que sea visible los alumnos.

- **Fecha y hora de fin**: con este parámetro el administrador indica la fecha y hora hasta la cual la competición estará activa. Una vez finalizado este plazo la competición pasará a estar inactiva.

- **Casilla de grupo magistral**: se trata de un elemento del tipo "checkbox" de HTML. Si se activa esta casilla en el formulario el grupo se tomará como grupo magistral y la competición se creará para todos los alumnos de los grupos reducidos que dependan del grupo magistral indicado previamente.

En este punto, antes de proceder a la creación de competición, se realiza la comprobación de los tiempos ingresados por el administrador en el formulario. No se puede especificar una fecha (entendida como fecha y hora) y tampoco una fecha de finalización previa a la de inicio. El sistema en este caso volverá al formulario y mostrará un mensaje de error.

Si se cumplen todas las condiciones, en este punto se procede a la creación de la competición y sus tablas a la función dinámica **create_competition_tables().**

Se dice que la función **create_competition_tables()** es dinámica porque no siempre crea las mismas tablas si no que crea, de acuerdo a los parámetros introducidos, una tabla distinta cada vez. Esta función se puede dividir en 3 partes. Estas partes se pueden ver en la figura 3.24 y se explicarán a continuación.

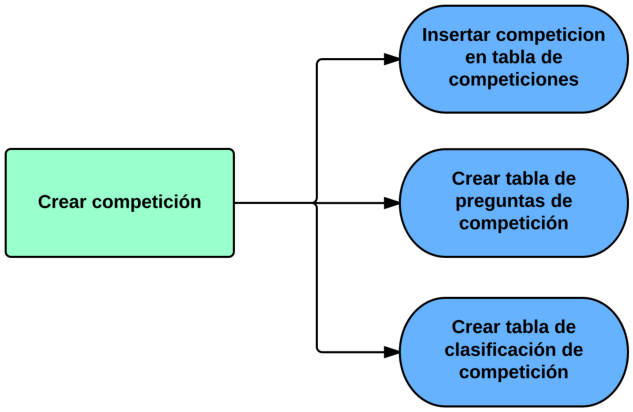


Figura 3.24: esquema de la operación create_competition_tables().

1) **Inclusión de la competición**: se llama a la función **insert_competition()** para ingresar en la tabla de competiciones la nueva competición con los datos especificados en el formulario.

2) **Creación de la tabla de preguntas**: la creación de la tabla de preguntas, a su vez, se realiza en tres fases. Se aporta la figura 3.25 como apoyo y se explican las 3 fases a continuación.

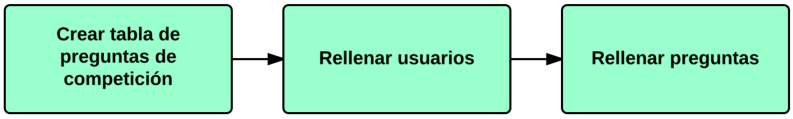


Figura 3.25: diagrama con las fases de la creación de la tabla de preguntas.

   a. **Creación de la tabla de preguntas de competición**: se crea una tabla vacía, como ya se explicó previamente en el apartado del modelo de datos, con el nombre añadiéndole la cadena de caracteres "_questions_table". En este momento, la competición está creada, pero no tiene ni usuarios ni preguntas asociados. Esta acción es llevada a cabo por la función **create_questions_table().**

   b. **Inclusión de los usuarios**: primero se llama a la función **get_group_list()** con el parámetro del formulario que especificaba el grupo. Esta función devuelve el listado de alumnos del grupo elegido y distingue entre magistral y reducido. Una vez tenemos la lista de identificadores de los usuarios se añade una fila por cada usuario y se rellena la primera columna de esa fila con el id de usuario. Este proceso es realizado por la operación **add_table_user().**

c. **Inclusión de las preguntas**: primero se llama a la función **get_random_questions().** Esta función, con los datos del formulario, obtiene los temas de los que se sacarán las preguntas de la tabla de preguntas (la cadena de temas que se ingresó en el formulario se divide con la función **explode()** de PHP) y devuelve una lista con los identificadores de pregunta que pertenezcan a los temas especificados. Estos identificadores están desordenados y se desordenan cada vez que se llama a la función. En esta parte del programa es donde se introduce la aleatoriedad de las preguntas para que no se repitan las mismas en todas las competiciones.

No se considera que sea malo que se repitan las preguntas en distintas competiciones, pues los estudiantes, por norma general, suelen tener que leer la teoría más de una vez y las preguntas tipo test no son una excepción. Por esto si que existe probabilidad de que se repitan preguntas pero con este sistema se asegura que no se repitan siempre las mismas. Mientras más preguntas por tema incluya el administrador menos probabilidad de repetición habrá.

Una vez se tiene la lista de identificadores desordenados se procede a añadir cada identificador de pregunta a la tabla como una nueva columna hasta cumplir el número de preguntas especificado por el administrador en el formulario o hasta que se acaben las preguntas. Esto es llevado a cabo por la operación **add_table_question()**.

3) **Creación de la tabla de clasificación**: de la misma manera que se crea la tabla de preguntas para la competición se crea la tabla de clasificación para la competición. Dado que en el apartado b. de la creación de la tabla de preguntas se obtiene un listado de todos los usuarios de la competición, en ese momento se crea la tabla de clasificación de competición y se le añade como segunda columna la de puntuación. Esto es llevado a cabo por la función **create_classification_table().**

En caso de que no se pueda crear alguna de las dos tablas o no se pueda añadir a la tabla de competiciones la información de la competición que se ha querido crear se notificará al administrador con un mensaje de error y se borrará toda la actividad del intento fallido de creación de competición, es decir,

la base de datos estará como antes de intentar crear la competición (la tabla de competiciones con las que tenía previamente y si se llegó a crear alguna de las tablas de la competición nueva se eliminarán). Esta acción es llevada a cabo por la función **delete_competition().** Esta función, además de borrar la entrada en la tabla de competiciones, llama a las funciones **delete_competition_questions_table()** y **delete_competition_classification_table().**

2- **Modificar competición:** el diagrama de flujo de esta operación se muestra a continuación:

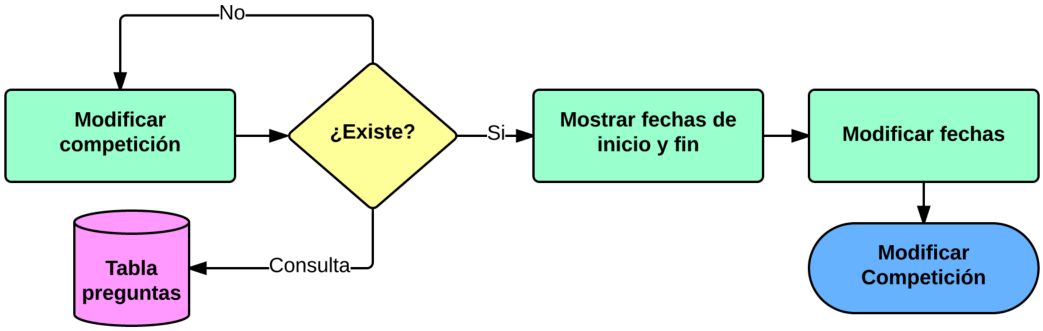


Figura 3.26: diagrama de flujo de la operación "Modificar competición".

Esta operación primero pregunta por pantalla el nombre de la competición que se desea modificar. Si la competición existe muestra los parámetros de fecha de inicio y fecha de fin de las competiciones y un formulario para su modificación. Igual que en la operación de creación de tabla, no se puede ingresar una fecha anterior a la actual ni tampoco una fecha de fin menor la de inicio.

Solo se podrán modificar los parámetros de las fechas de la competición porque el resto de parámetros intervienen en la creación de las tablas asociadas y no es posible su modificación. Si quisiésemos modificar los otros parámetros de la competición habría que eliminarla y crearla de nuevo con el mismo nombre y distintos parámetros. Si se cumplen todas las condiciones, la modificación de los tiempos lo llevará a cabo la función **modify_competition_dates().**

En caso de que se deje algún parámetro de los formularios en blanco o se inserten incorrectamente las fechas se notificará al administrador con un mensaje de error.

3- **Mostrar competiciones:** esta operación es, al igual que las otras dos operaciones "Mostrar" del sistema una operación de carácter consultivo a la base de datos.

Muestra por pantalla una tabla con todos las filas y columnas de la tabla de competiciones. Esta consulta es llevada a cabo por la función **show_competitions()**.

4- **Borrar competición:** el diagrama de flujo de esta operación se muestra en la figura 3.27

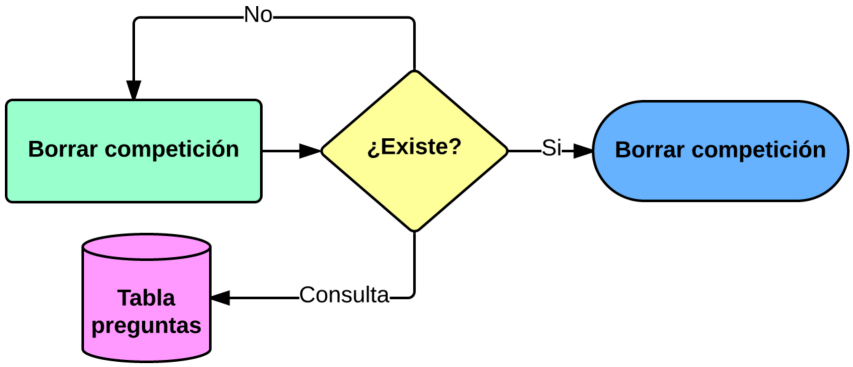


Figura 3.27: diagrama de flujo de la operación "Borrar competición".

Esta operación despliega por pantalla una lista con las competiciones existentes y pregunta al administrador mediante un formulario el nombre de la competición que desea borrar. Si el nombre es correcto, se elimina la competición así como todas sus tablas asociadas. Esta acción es llevada a cabo por la función **delete_competition()** que se explicó en la tabla y en la operación "Crear competición".

En caso de que no se pueda borrar la competición, no se haya introducido el nombre o éste sea inválido se notifica un mensaje de error al administrador.

3.3.3.4 Clasificación.

La operación de clasificación es la ultima de las opciones disponibles para el administrador, aunque también está disponible para los alumnos. Su diagrama de flujo puede verse en la figura 3.28.

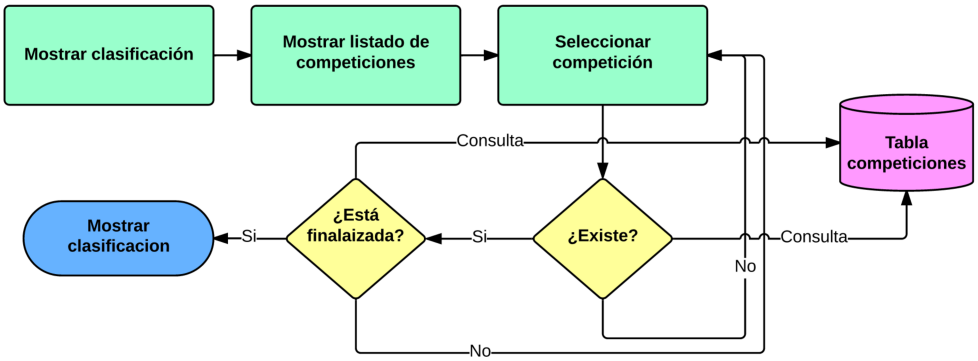


Figura 3.28: diagrama de flujo de la operación "Clasificación".

Esta operación primero desplegará una lista con todas las competiciones que ya hayan terminado y preguntará al usuario cual es la competición de la que desea ver los resultados.

El sistema de clasificación que se ha descrito es sencillo: Al configurar las preguntas de la competición en la tabla de preguntas, cada jugador tendrá realizar la competición en el plazo establecido. Para cada respuesta se asignara una puntuación de acuerdo al nivel de la pregunta. Si la pregunta es de nivel 1 será un punto, si es nivel 2 serán dos puntos y así sucesivamente.

Una vez acabado el plazo de la competición se hará accesible la tabla de clasificación de dicha competición. Donde cada usuario tendrá la suma de la puntuación de cada pregunta y se desplegará la tabla ordenada con los identificadores de usuario, su puntuación y su puesto. Esta acción es llevada a cabo por la función **show_classification().** Las comprobaciones se efectúan con la función **get_competition_times().**

En caso de que no se seleccione ninguna competición, se seleccione una errónea o no se pueda mostrar la clasificación se mostrará un mensaje de error.

Operaciones de Alumnos.

Las operaciones de los alumnos son dos: Realizar competición y mostrar clasificación.

1- **Mostrar clasificación**: esta operación es igual que la operación de "Mostrar clasificación" de la administración, pero solo permite elegir al alumno las competiciones en las que el participa.

2- **Realizar competición**: el diagrama de flujo de esta operación se muestra en la figura 3.29.

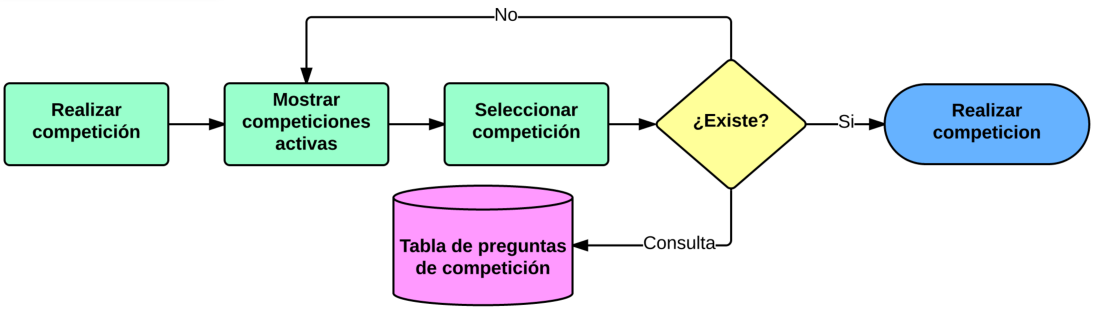


Figura 3.29: diagrama de flujo de la operación "Realizar competición".

En esta operación al alumno se le muestran las competiciones que están activas y que no ha realizado hasta el momento. El alumno podrá elegir una de entre las mostradas y proceder a realizarla. Para mostrar las competiciones activas se obtiene, mediante la función **get_group_for_user()** el grupo del alumno que está solicitando las competiciones que tiene disponibles. De acuerdo a ese grupo se utilizan después las funciones **get_competitions_reducido()** y **get_competitions_magistral()**, que devolverán , cada una, una lista de los nombres de las competiciones que afecten a los grupos magistral y reducido del alumno.

Una vez obtenidas todas las competiciones se procede a la comprobación de fechas. Solo se mostrarán por pantalla a los alumnos las competiciones que están dentro del rango de fechas que define la competición, es decir, si la fecha de la solicitud en el momento de hacerla es mayor que la fecha de inicio y menor que la fecha de fin de la competición. Los datos de fecha de inicio y fecha de fin para cada una de las competiciones de las dos listas se obtienen con llamadas sucesivas a la función **get_competition_times().**

En caso de error al ingresar el nombre de la competición se devuelve un mensaje de error al alumno y se vuelven a mostrar las competiciones activas para que el alumno proceda de nuevo a la elección.

Además de la comprobación de tiempos se comprueba también que la competición no esté ya realizada, con el fin de que los alumnos no puedan modificar sus resultados después de contestar las preguntas. Esto se realiza con la función **check_actual_question().** En el caso de que algún alumno se hubiese quedado a mitad de la competición por algún tipo de problema (por ejemplo, un

problema de conexión) la competición se reanudaría en la siguiente pregunta que le correspondiese al alumno.

Una vez elegida la competición se procede a su realización. El flujo de la competición se muestra en el diagrama 3.30 y se explica a continuación.

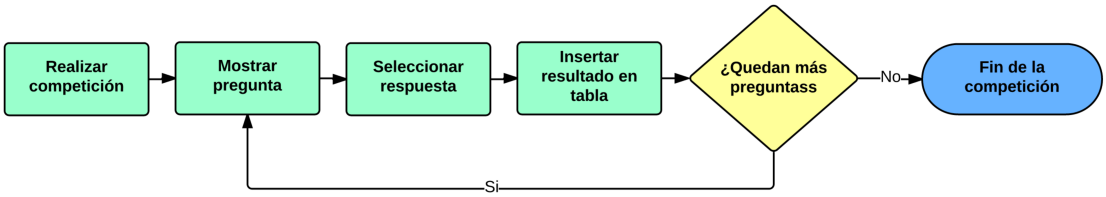


Figura 3.30: diagrama de flujo de la realización de una competición.

Una vez se selecciona una competición se obtiene una lista de los identificadores de todas las preguntas que tiene dicha competición para posteriormente ir cargándolas una a una. Esta consulta a la tabla se realiza mediante la función **get_competition_questions().**

A continuación se despliegan por pantalla, de una en una, las preguntas de la competición con sus 4 respuestas posibles dentro de un formulario HTML. La operación con la que se obtienen los datos de la pregunta para su presentación por pantalla es **get_question()**. Cada una de las respuestas tiene asociada un botón tipo "radio" de HTML para elegir la opción que se considere correcta, y una vez enviado el resultado se procede a la inserción de la puntuación en la tabla.

El calculo de la puntuación es muy sencillo. Se multiplica el nivel de la pregunta por 1 y se inserta dicho valor en la tabla correspondiente a la competición, fila correspondiente al usuario y columna correspondiente a la pregunta. Este proceso se repite hasta que se han realizado todas las preguntas del test.

En todas las operaciones, tanto de administración como de alumnos, al igual que se han incluido los mensajes de error, cuando se termina de realizar cualquiera de las operaciones de manera correcta se ha incluido un mensaje de "Operación realizada correctamente".

3- **Cambiar contraseña:** el diagrama de flujo de esta operación se muestra en la figura 3.31.

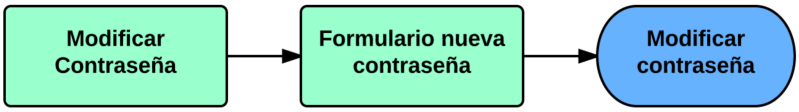


Figura 3.31: diagrama de flujo de la operación "Cambiar contraseña".

Esta operación permite al alumno cambiar su contraseña. Se le muestra un doble formulario para que inserte la nueva contraseña y su verificación y, si es correcto, se modifique la contraseña. Esta modificación se realiza con la función **modify_user().**

En caso de error, ya sea porque no se haya podido modificar la contraseña, porque el usuario no haya rellenado el formulario o las contraseñas no coincidan, se muestra un mensaje de error.

Capitulo 4: Validación de requisitos.

En esta sección del documento detallaremos las pruebas que se han realizado a la aplicación con el fin de validar los requisitos que se detallaron en la tabla 3.2. Para este propósito, además de explicar las pruebas, se incluirán capturas de pantalla de apoyo cuando sea útil para las comprobaciones. Se procede, a continuación, a hacer un listado con todos los requisitos:

4.1. Validación.

En esta sección se procede a la validación de los requisitos uno a uno, explicando por qué se validan en el diseño y la implementación en algunos casos, y las pruebas que se han realizado para la comprobación en otros.

Requisitos generales:

- **Requisito # 1:** el requisito de existencia de dos tipos de usuario: Alumno y administrador se cumple a lo largo del diseño y la implementación así como a lo largo de la cumplimentación del resto de requisitos. Esto es así porque la aplicación ha sido diseñada de manera que existe una diferenciación clara y muy definida entre los dos tipos de usuarios permitiendo a cada uno solo las operaciones que le corresponden.

- **Requisito # 2:** a lo largo del diseño y la implementación de todas las funcionalidades surgieron a partir de la tabla de requisitos se han utilizado todos los parámetros que se incluyeron en la tabla de usuarios en la base de datos y no fue necesario la inclusión de ninguno nuevo.

- **Requisito # 3:** este requisito se cumple en el diseño de la base de datos, pues todos los datos de los usuarios se han guardado en la tabla de usuarios de la base de datos.

- **Requisito # 4:** como se explicó en la operación “Registro de usuarios” la contraseña de usuario se ha guardado en la tabla de usuarios de la base de datos cifrada con la función MD5() de PHP. Se puede ver la contraseña cifrada de longitud 32 bytes en la figura 4.1.

id	email	password	magistral
100072644	100072644@alumnos.uc3m.es	7c6a180b36896a0a8c02787eeafb0e4c	95
100072645	100072645@alumnos.uc3m.es	6cb75f652a9b52798eb6cf2201057c73	95
100072646	100072646@alumnos.uc3m.es	819b0643d6b89dc9b579fdfc9094f28e	91
100072647	100072647@alumnos.uc3m.es	819b0643d6b89dc9b579fdfc9094f28e	95

Figura 4.1: captura de tabla de usuarios en terminal MySQLl.

- **Requisito # 5:** este requisito, al igual que el requisito #1, se cumple a lo largo de toda la implementación. En todas las páginas web que componen la aplicación se ha incluido la comprobación de las variables de sesión para mantener la sesión del navegador iniciada en todo momento. Además con el uso de sesiones se ha implementado también la estructura de control de acceso a las distintas partes de la aplicación.

Requisitos de operaciones de administración:

La figura 4.2 muestra una captura de pantalla del menú de administrador de la aplicación.

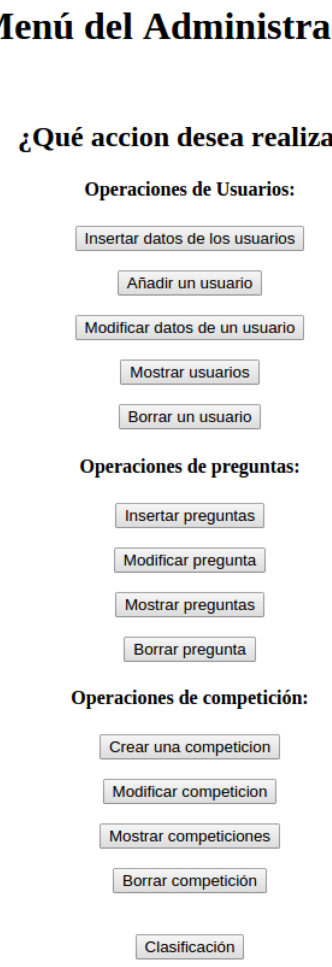


Figura 4.2. visión del menú de administrador.

- **Requisito # 6:** la operación de registrar usuarios solicita a través de un formulario el archivo CSV para la inclusión de los usuarios en la tabla de usuarios. Utilizamos la operación "Mostrar usuarios" antes y después de la inclusión de los usuarios para validar este requisito y, de paso, el **requisito #9**. La primera captura se corresponde con el formulario de petición de archivo CSV y las capturas con las que se validarán el requisito serán las figuras 4.3 y 4.4.

Id	Nombre	Primer Apellido	Segundo Apellido	Correo Electrónico	Contraseña	Grupo Magistral	Grupo Reducido	Permisos
1	Iria	Estevez	Ayres	admin@gmail.com	81dc9bdb52d04dc20036dbd8313ed055	1	1	1

Figura 4.3: operación "Mostrar usuarios" antes de operación "Registrar Usuarios".

Id	Nombre	Primer Apellido	Segundo Apellido	Correo Electrónico	Contraseña	Grupo Magistral	Grupo Reducido	Permisos
1	Iria	Estevez	Ayres	admin@gmail.com	81dc9bdb52d04dc20036dbd8313ed055	1	1	1
100072644	Ignacio	Ortiz	Luengo	100072644@alumnos.uc3m.es	7c6a180b36896a0a8c02787eeafb0e4c	95	95	0
100072645	Usuario1	Apellido1 usuario1	Apellido2 usuario1	100072645@alumnos.uc3m.es	6cb75f652a9b52798eb6cf2201057c73	95	95	0
100072646	Usuario2	Apellido1 usuario2	Apellido2 usuario2	100072646@alumnos.uc3m.es	819b0643d6b89dc9b579fdfc9094f28e	91	91	0
100072647	Usuario3	Apellido1 usuario3	Apellido2 usuario3	100072647@alumnos.uc3m.es	819b0643d6b89dc9b579fdfc9094f28e	95	92	0

Figura 4.4: operación "Mostrar usuarios" después de operación "Registrar Usuarios".

Se ha comprobado a intentar registrar los usuarios desde el mismo archivo CSV para comprobar así que el sistema comprueba correctamente que los usuarios ya están registrados y no los inserta en la tabla.

- **Requisito # 7:** la operación "Registrar usuario" funciona de la misma manera que la operación "Registrar usuarios" del requisito #6. Se ha procedido a registrar un usuario a través de formulario con resultado satisfactorio.

- **Requisito # 8:** para validar este requisito, correspondiente a la operación "Modificar usuario" se ha procedido a modificar un usuario previamente introducido. La operación funciona correctamente.

- **Requisito # 9:** validado en el **requisito # 6**.

- **Requisito # 10:** para validar este requisito, correspondiente a la operación "Borrar usuario" se ha procedido a eliminar un usuario previamente introducido. La operación funciona correctamente.

- **Requisito # 11:** el proceso de comprobación de este requisito, correspondiente a la operación "Subir preguntas" ha sido el mismo que el seguido con el requisito #6. La comprobación ha sido correcta.

- **Requisito # 12:** para validar este requisito, correspondiente a la operación "Modificar pregunta", se ha modificado una pregunta previamente introducida en el sistema. La operación funciona correctamente.

- **Requisito # 13:** se ha hecho una captura de pantalla para mostrar que este requisito, correspondiente a la operación "Mostrar preguntas" se cumple. Se trata de la figura 4.5.

Id	Tema	Enunciado	Respuesta 1	Respuesta 2	Respuesta 3	Respuesta 4	Respuesta correcta	Nivel
22	5	Considera el siguiente fragmento de código: struct data { struct data *s; } a, b, c; ¿Qué tres líneas de código son necesarias para crear una estructura de datos circular (a.s apunta a b, b.s apunta a c, y c.s apunta a a)?	a.s = b; b.s = c; c.s = a;	&a.s = b; &b.s = c; &c.s = a;	a.s = *b; b.s = *c; c.s = *a;	a.s = &b; b.s = &c; c.s = &a;	4	5
23	5	Considera el siguiente fragmento de código: struct data { int i; int j; } a; struct data *b = &a; ¿Cómo se puede calcular la suma de los campos i y j de la variable a utilizando sólo la variable b?	b->i + b->j	*b->i *b->j	b.i b.j	No se puede calcular.	1	5
24	5	El puntero a contiene la dirección de memoria del puntero b que contiene la dirección de memoria del entero c. ¿Cual de las siguientes expresiones le asigna el valor 30 al entero c?	*a = 30	**a = 30	***a = 30	No se puede hacer esa asignación	2	5

Figura 4.5: captura de pantalla de la operación "Mostrar preguntas".

- **Requisito # 14:** para validar este requisito, correspondiente a la operación "Borrar pregunta" se ha procedido a eliminar una pregunta previamente introducida. La operación funciona correctamente.

- **Requisito # 15:** para comprobar este requisito, correspondiente a la operación "Crear competición" se ha procedido a crear una competición con los parámetros adecuados y se ha realizado después la operación correspondiente al requisito # 16 para comprobar que se ha creado correctamente. La figura 4.6 lo corrobora, pues hay varias competiciones creadas. Con esta comprobación también se comprueba el **requisito # 17**, pues se realiza una operación de "Mostrar competiciones" para la comprobación.

Nombre	Temas	Grupos	# Preguntas	Fecha de inicio	Fecha de fin
test	1,5,7	91*	11	03/17/2015 09:00:00	03/17/2016 09:01:00
test2	1,5,7	92	7	03/17/2015 09:00:00	02/11/2016 21:30:00
test3	1,5,7	91*	11	03/17/2015 09:00:00	03/17/2016 09:01:00
tes4	1,5,7	91	11	03/17/2015 09:00:00	02/11/2016 21:30:00
competicion1	1,2,3,4,5	95	5	02/20/2015 12:24:00	02/11/2016 21:30:00

Figura 4.6: operación "Mostrar competiciones" después de la operación "Crear Competición".

- **Requisito # 16:** para validar este requisito, correspondiente a la operación "Modificar competición" se ha procedido a modificar los tiempos de una competición ya creada obteniendo un resultado satisfactorio.

- **Requisito # 17:** validado en el requisito #15.

- **Requisito # 18:** para la validación de este requisito se procedió a borrar una competición previamente introducida con resultado satisfactorio.

- **Requisito # 19:** para la validación de este requisito fue necesario realizar una competición con varios usuarios distintos. Una vez acabado el plazo de competición se accedió a la clasificación de la competición con uno de los usuarios participantes y se visualizó la tabla de clasificación de la competición. Como soporte la figura 4.7. Adicionalmente, se realizó una consulta por la terminal de la tabla de preguntas de competición (ver figura 4.8) para comprobar que la clasificación fue calculada correctamente. El resultado fue satisfactorio.

Clasificación de la competición "competicion1":

ID	Puntos totales
100072644	5
100072645	4
100072658	0

Figura 4.7: captura de tabla de clasificación de la competición.

```
mysql> select * from  competicion1_questions_table;
+-----------+------+------+------+------+------+
| id        | 24   | 26   | 25   | 23   | 29   |
+-----------+------+------+------+------+------+
| 100072644 |    5 |    0 |    0 |    0 |    0 |
| 100072645 |    0 |    1 |    1 |    0 |    1 |
| 100072658 | NULL | NULL | NULL | NULL | NULL |
+-----------+------+------+------+------+------+
```

Figura 4.8: captura de consulta de la tabla de preguntas de competición.

Requisitos de operaciones de usuarios:

- **Requisito # 20:** para la comprobación de este requisito se ha efectuado la operación de usuario "Modificar contraseña", se ha salido de la aplicación y se ha vuelto a logear el usuario en la aplicación con su nueva contraseña de manera satisfactoria.

- **Requisito # 21:** para validar este requisito se ha seguido el proceso de competición del usuario con identificador 100072658 de la competición anterior. Se procede a presentar una secuencia de capturas de pantalla en las que se ve el proceso . Se muestran también una captura de la tabla de preguntas de competición y la clasificación de la competición para comprobar que se han añadido correctamente los resultados a la tabla de clasificación.

  En las figuras a continuación se puede observar la secuencia de operaciones realizadas para la comprobación.

Figura 4.9: visión del menú del usuario.

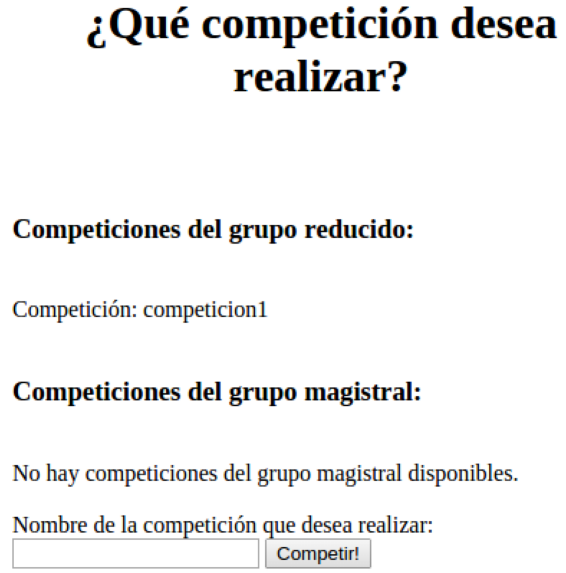


Figura 4.10: captura de selección de competición.

pregunta numero 1:

El puntero a contiene la dirección de memoria del puntero b que contiene la dirección de memoria del entero c. ¿Cual de las siguientes expresiones le asigna el valor 30 al entero c?

- *a = 30
- **a = 30
- ***a = 30
- No se puede hacer esa asignación

Siguiente pregunta

Figura 4.11: ejemplo selección de respuesta a pregunta.

Clasificación de la competición "competicion1":

ID	Puntos totales
100072658	6
100072644	5
100072645	4

Figura 4.12: captura de clasificación final de la competición.

```
mysql> select * from  competicion1_questions_table;
+-----------+------+------+------+------+------+
| id        | 24   | 26   | 25   | 23   | 29   |
+-----------+------+------+------+------+------+
| 100072644 |    5 |    0 |    0 |    0 |    0 |
| 100072645 |    0 |    1 |    1 |    0 |    1 |
| 100072658 |    5 |    1 |    0 |    0 |    0 |
+-----------+------+------+------+------+------+
```

Figura 4.13: captura de pantalla de la tabla de preguntas de competición.

- **Requisito # 22:** requisito ya comprobado en la comprobación del **requisito # 19**.

Requisitos de Administración:

- **Requisito # 23:** cuando se realizó la operación “Registrar usuarios” para comprobar el **requisito #6** se añadieron varios usuarios desde el archivo CSV, por lo que este requisito está validado.

- **Requisito # 24:** de igual manera, al validar el **requisito #7** se comprobó que funcionaba de esta manera.

- **Requisito # 25:** con la captura de pantalla de la figura 4.14 vemos que el requisito de “Modificar usuario” se cumple.

Figura 4.14: comprobación de la operación "Modificar Usuario".

- **Requisito # 26:** validado en el **requisito #9**.

- **Requisito # 27:** validado en el **requisito #10**.

- **Requisito # 28:** este requisito se asegura durante el diseño y la implementación, ya que al diseñar la base de datos toda la información referente a las preguntas se guarda en la tabla de preguntas y en la implementación se ha asegurado que las operaciones referentes a las preguntas sean consistentes con este requisito.

- **Requisito # 29:** validado en el **requisito #11**.

- **Requisito # 30:** para validar este requisito se ha procedido a modificar los datos de una pregunta elegida de entre la lista a través del formulario mostrado con resultado satisfactorio.

- **Requisito # 31:** en el **requisito #13** se ha comprobado que la operación "Mostrar preguntas" ha mostrado por pantalla todas las preguntas de la base de datos que corresponden al tema 5.

- **Requisito # 32:** validado en el **requisito #14**.

- **Requisito # 33:** este requisito también se asegura durante el diseño y la implementación, ya que al diseñar la base de datos toda la información referente a las competiciones se guarda en la tabla de preguntas y en la

implementación se ha asegurado que las operaciones referentes a las preguntas sean consistentes con este requisito.

- **Requisito # 34:** validado en el **requisito #15**.

- **Requisito # 35:** para comprobar este requisito se ha procedido a indicar una competición en el formulario de petición y a introducir unos nuevos valores de tiempo. El resultado obtenido ha sido el deseado.

- **Requisito # 36:** validado en el **requisito #17** (que, a su vez, se valida en el **requisito #15**).

- **Requisito # 37:** requisito validado en el **requisito # 19**.

Requisitos de usuarios:

- **Requisito # 38:** validado en el **requisito #20.**

- **Requisito # 39:** para comprobar que se muestra más de una competición si hay varias activas y de grupos magistral y reducido distintos se ha procedido a crear 5 competiciones que afectasen a un alumno perteneciente al grupo 91, magistral y reducido, de las cuales 3 estén activas y 2 sean para un comienzo futuro, de manera que, por pantalla, solo pueda elegir entre 3 competiciones. El funcionamiento obtenido fue el deseado. Se adjunta la captura de pantalla en la figura 4.15, donde se muestran las competiciones disponibles para dicho usuario del grupo 91.

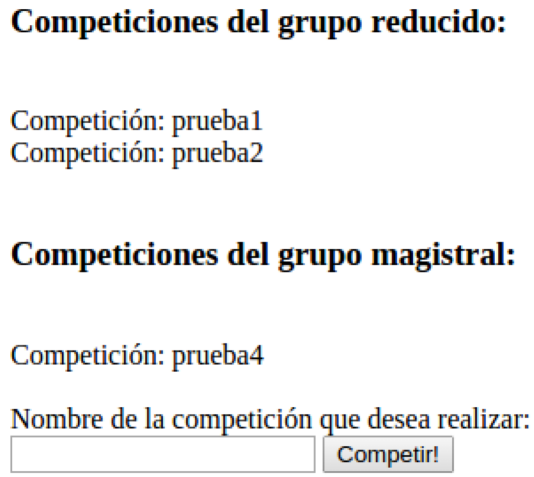


Figura 4.15: demostración de las competiciones disponibles y activas de un usuario.

- **Requisito # 40:** validado en el **requisito #19.**

4.2. Conclusiones de la validación de los requisitos.

Tras realizar todas las pruebas para la comprobación de todos los requisitos se concluye que la aplicación cumple con los objetivos de diseño que se plantearon antes de la implementación.

Capitulo 5: Conclusiones y trabajo futuro.

En esta sección se procede a exponer las conclusiones que se han obtenido de la realización de este trabajo de fin de grado así como una serie de ideas para el desarrollo futuro de la aplicación.

5.1. Conclusiones.

El desarrollo de esta aplicación web ha servido para aprender una forma de diseñar e implementar una aplicación web basada en el modelo cliente-servidor.

En esta aplicación se ha implementado un sistema bajo los requerimientos de un profesor de la universidad con el fin de que sirva como plataforma para la realización de competiciones de preguntas tipo test y con todas las funcionalidades que se han creído necesarias.

El empleo de archivos CSV para la subida de información a la aplicación así como la versatilidad de la que se ha dotado al administrador para modificar la información del sistema hace que esta aplicación cumpla las expectativas teóricas. Será en un entorno real donde se demuestre la completa funcionalidad del sistema.

También se ha comprendido la necesidad de realizar un estudio tecnológico inicial. El haber estudiado el estado del arte de las aplicaciones web ha dado una visión general mucho más amplia de las tecnologías existentes para el desarrollo de aplicaciones web.

Por ultimo se valora positivamente el desarrollo de un trabajo de estas dimensiones, nunca antes realizado durante la carrera con el que se aprende y se ponen en común tantos conceptos de la carrera y se ve la importancia de muchos aspectos hasta el momento considerados menos importantes.

5.2. Trabajo futuro.

Esta aplicación para el trabajo de fin de grado puede ser solo el comienzo de una aplicación que se puede seguir desarrollando además de, como ya se ha comentado, exportarla a una plataforma móvil.

Algunas ideas para la mejora de la aplicación son:

- Incorporación de nuevas dinámicas de competición: se mejoraría la dinámica competitiva de los alumnos con la implantación en la aplicación de un sistema de duelos, en la que alumnos puedan retarse entre si en un modelo de competición clasificatorio o por eliminatorias.

- Añadir funcionalidades estadísticas: distintas formas de mostrar la clasificación, comparación de los alumnos de distintos grupos o una comparativa de la progresión a lo largo de las competiciones a lo largo del curso son solo unas pocas ideas de las muchas funcionalidades que se pueden añadir.

- Crear una interfaz de usuario más amigable, tanto para administrador como para el alumno.

Capitulo 6:Planificación y presupuesto.

6.1. Planificación.

Este trabajo de fin de grado ha sido realizado durante 4. En la siguiente tabla se especifican las distintas tareas llevadas a cabo y el número de horas empleadas en cada una de ellas.

# Tarea	Explicación	# Horas
1	Estudio tecnológico	**50**
2	Instalación del entorno LAMP	**5**
3	Aprendizaje HTML y PHP	**45**
4	Aprendizaje de bases de datos y MySQL	**50**
5	Diseño de la aplicación	**25**
6	Diseño de la base de datos	**50**
7	Programación de la aplicación	**100**
8	Validación	**25**
9	Memoria	**70**
	Total horas:	**420**

Figura 6.1: tabla de tareas.

Se ha dividido el proyecto en 9 tareas distintas que se explican a continuación.

1- **Estudio tecnológico**: en esta tarea correspondo al inicio del proyecto. Se dedicaron estas 75 horas de trabajo al estudio del concepto de aplicación web, los distintos tipos que existen y al análisis de las tecnologías que se usarían.

2- **Instalación del entorno LAMP**: para desarrollar la aplicación y, previamente, adquirir los conocimientos necesarios para dicho desarrollo se instaló y configuró el entorno.

3- **Aprendizaje HTML y PHP:** en esta actividad se leyeron tutoriales para estudiar estas dos tecnologías y se hicieron pruebas de dificultad creciente con el objetivo de aprender el lenguaje.

4- **Aprendizaje de bases de datos y MySQL**: durante esta actividad se realizaron lecturas de un tutorial de SQL así como de uno de MySQL. Como en el caso anterior, se fueron realizando diversas pruebas para fijar los conceptos de la lectura.

5- **Diseño de la aplicación**: una vez adquiridos los conceptos necesarios se procedió al diseño de la aplicación. Durante esta tarea se plantearon los requisitos y se diseñó cómo iba a programarse y cumplirse cada uno de los requisitos.

6- **Diseño de la base de datos**: después del diseño de la aplicación se diseñaron las tablas de la base de datos para dar soporte a los datos de la aplicación.

7- **Programación de la aplicación**: durante esta fase del proyecto se implementó lo diseñado durante las dos tareas anteriores.

8- **Validación**: durante esta fase se realizaron las pruebas de validación de requisitos así como las pruebas de generación de errores de la aplicación.

9- **Memoria**: la última tarea del proyecto ha sido la realización de ésta memoria.

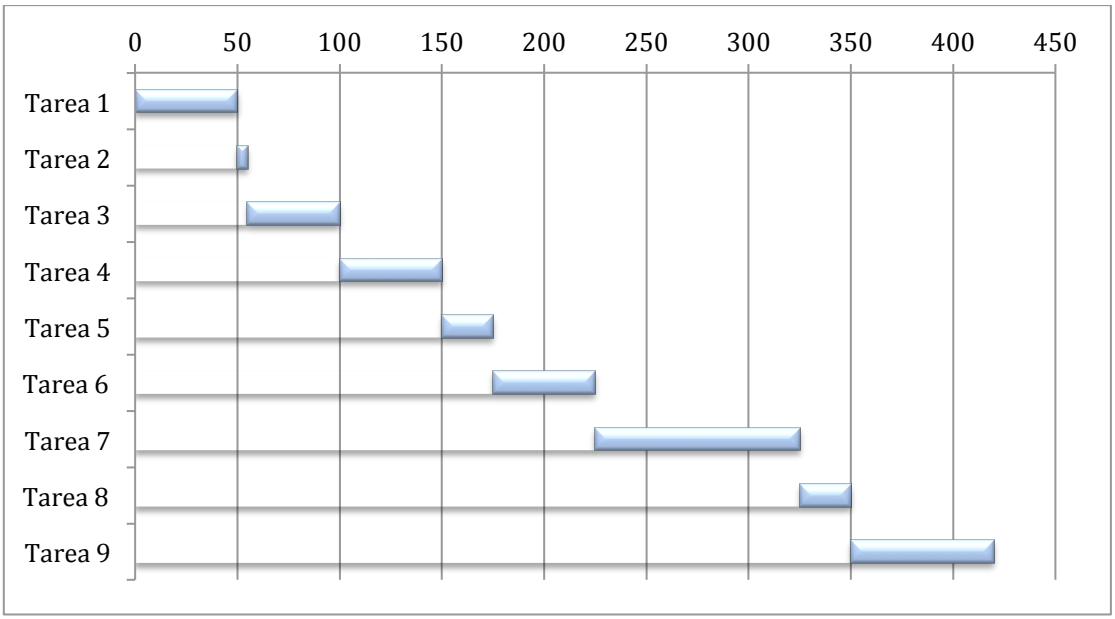


Figura 6.2: diagrama de Gantt.

6.2. Presupuesto.

En esta sección se presenta el presupuesto económico para la realización de la aplicación web. Se han tenido en cuenta las horas totales empleadas en la realización de la aplicación, el coste económico del material empleado así como los gastos de servicios.

En cuanto a las horas empleadas en el desarrollo se ha utilizado como número el número de horas de creación de la aplicación (se han excluido las horas de aprendizaje). Se ha calculado sumando las horas de las tareas 5, 6, 7 y 8 de la sección anterior. Los honorarios por hora de trabajo del desarrollador han sido de 25 euros.

Los gastos de material corresponden a un ordenador de gama media y el libro utilizado para el las consultas y el aprendizaje.

Se resume el presupuesto del proyecto en la siguiente tabla:

	Cantidad	Coste	Total
Horas de trabajo	200	25,00€/h	5000,00€
Ordenador gama media	1	700,00 €	700,00 €
Libros para documentación	1	30,00€	30,00€
Subtotal			5730,00€
IVA	21%		1209’60€
TOTAL			6933’30€

Figura 6.3: presupuesto de la aplicación.

Capitulo 7:Bibliografía.

[1] *http://www.agpd.es* Consultado 5-Enero-2014
[2] *http://www.utdallas.edu/~chung/SA/2client.pdf* Consultado 1-Noviembre-2014
[3] *http://www.iis.net/* Consultado 5-Noviembre-2014
[4] *www.nginx.org* Consultado 5-Noviembre-2014
[5] *http://www.apache.org/* Consultado 5-Noviembre-2014
[6] *http://www.postgresql.org/* Consultado 11-Noviembre-2014
[7] *http://www.oracle.com/es/index.html* Consultado 12-Noviembre-2014
[8] *http://www.microsoft.com/sqlserver/* Consultado 12-Noviembre-2014
[9] *http://www.mysql.com/* Consultado 3-Noviembre-2014
[10] *http://www.asp.net/* Consultado 18-Noviembre-2014
[11] *https://www.python.org/* Consultado 18-Noviembre-2014
[12] *http://www.oracle.com/technetwork/java/javaee/servlet/index.html* Consultado 14-Noviembre-2014
[13] *www.php.net* Consultado 3-Noviembre-2014
- http://php.net/manual/es/ref.mysql.php Consultado 16-Diciembre-2014
[14] "*Learning PHP, MySQL, Javascript, CSS & HTML5*". Tercera edición. Autor: Robin Nixon. Consultado 3-Noviembre-2014
[15] *http://www.desarrolloweb.com/articulos/instalar-php-apache-mysql-linux.html* Consultado 25-Noviembre-2014
[16] *http://www.yiiframework.com/doc/guide/1.1/en/basics.best-practices* Consultado 5-Noviembre-2014
[17] *http://www.w3schools.com/sql/* Consultado 15-Diciembre-2014
-
[18] *www.desarrolloweb.com* Consultado 1-Diciembre-2014
- http://www.desarrolloweb.com/manuales/12/ Consultado 1-Diciembre-2014
- http://www.desarrolloweb.com/html/ Consultado 1-Diciembre-2014
- http://www.desarrolloweb.com/manuales/34/ Consultado 13-Diciembre-2014
[19] *www.wikipedia.es* Consultado 3-Noviembre-2014
[20] www.stackoverflow.com Consultado 1-Diciembre-2014

Capitulo 8: Synopsis of the graduating thesis.

8.1. Introduction.

8.1.1. Synopsis.

Since the start of Bologna program, most of the careers' subjects have become courses with big amounts of work along the week. Consequently, this makes many students to drop the subjects during the continuous evaluation period.

As almost all the subjects contain a theoretical part, the idea of making a small game in a questions-multiple answers basis has become popular. This game will be an additional tool for both students and professors.

Students will benefit from a competition tool that will serve them in different ways. First of all, it will motivate them to study the different subjects. This increase in motivation will be achieved by competing several times along the semester, as a consequence this will generate a dynamic learning atmosphere different from the traditional one. Moreover, it will provide a reference for the level of the subject exams.

On the other hand, Professors will take advantage of this application in some other ways. Apart from the motivation that will be generated in the students, professors will know if the subjects they are teaching are being learnt in a proper way. Additionally, Professors will be able to reinforce the knowledge that the tests show is not properly learnt and that needs to be reviewed again. Finally, it can be used as an evaluation tool.

This graduate project could be the start for other students to develop, in the future, new competition modes inside the architecture that is going to be exposed in the next pages.

8.1.2. Objectives.

Having in mind the previously described considerations, the application has been designed and developed as a web application, based on a list of requirements that will serve the professors of the university to use these competitions. For this purpose the following objectives have been outlined:

- Carry out a theoretical analysis of the different technological options available to develop the project.
- Study the legal framework that affects this project.
- Design and implement all the functionalities needed for the correct functioning of the application.
- Create a solid architecture for improving the program in the future and exporting it to other platforms such as mobile devices.

8.1.3. Legal framework.

As this is an application that uses and deals with personal information of a group of users the legal framework has been researched. In Spain we have to look at the data protection law (LOPD) and to the the Spanish Data protection agency (AEPD), which is in charge of overseeing the compliance with the LOPD.

According to this law, the software developers are responsible of the protection of personal data. This law distinguishes between 3 levels of information:

- **Basic**: data such as name, last names, zip codes or address.
-
- **Medium:** professional data (Curriculum Vitae) or banking data.
-
- **High:** health care data or political information.
-

Once the information level has been assigned, the software developers should fill a form in order to receive the register the information to be stored. After this, the AEPD will provide a Register Identification Number to identify that the information should be protected.

Once it is obtained that identifier the document in Figure 1.3 should be filled with the data type that needs to be protected.

8.1.4. Thesis structure.

The thesis document is divided into the 6 following sections:

1- **Introduction**: already explained, describes the motivation of doing this project, as well as the objectives and legal framework that affects it.

2- **State of the art**: describes the actual technological solutions that are available for doing the project and explains the reason for the choices done.

3- **Design and implementation**: this section first explains the installation of the development environment. Afterwards, it lists the requirements and exposes the design and implementation followed by the developer in order to accomplish those requirements.

4- **Validation**: this section will cover all the tests done to the software to prove if the requirements are fulfilled.

5- **Conclusions and future work**: this section will cover the conclusions of the conducting this graduate project as well as some ideas for the future of the application

8.2. State of the art.

In order to make a good design and posterior development of the project a technological study has been conducted in order to choose the best solutions for the application.

It has been chosen a client-server architecture model. This is a distributed application model that is divided in two different parts: a server that supplies service and clients that requests services to the server. Both of these environments are connected through a net (usually internet).

In this model we have analysed multiple technological options for each different part of the program and in the following table the results are exposed. The reason of its election of each technology upon the others is given.

	Analysed technologies	Used in the application	Reason
Application Server	- Internet information services - Nginx - Apache	Apache	- The most stable. - Multiplatform. - Wide variety of already programmed libraries for multiple programming languages. - Easy to find tutorials and documentation.
Database	- PostgreSQL - Oracle database - Microsoft SQL server - MySQL	MySQL	- Low resources consumption. - Higher operating speed. - No economical cost. - More intuitive administration tools. - Faster than PostgreSQL.
Server side language	- Active Server Pages - Python - Java Server pages and Servlets - PHP	PHP	- No economical cost. - Highest speed. - Easy to find tutorials and documentation as its use is widely extended. - Very stable. - Large number of libraries.

For the client side it was decided to use HTML as the objective of the project was not the development of a user-friendly environment but the creation of an application that provides all the necessary functionalities.

8.3. Design and implementation.

After the setup of the working environment LAMP, the requirement list was created. In this list of 39 requirements all the needs of the application were exposed. The requirements of the application are summarized in the following list:

- There must exist two types of users: student and professor, each of them with privileges for the correct administration of the system.

- Each type of user should have all the necessary information and it should be stored in the database.

- The questions should have all the necessary information to be answered and corrected later in the competitions as well as the topic they belong to.

- Passwords must be ciphered.

- User sessions should be used.

- Professors should be able to: register users (from "comma separated values (CSV) file and individually), modify users data, show user list, delete users, upload questions (from CSV file too), modify question data, show questions, delete questions, create competitions, modify competitions, show competitions, delete competitions and show classification

- Students should be able to: change password, make competition and show classification.

- Database should store the information in different tables:
    - Users information: all the information needed for the users' administration.
    - Questions information: all the information needed for the questions' administration.
    - Competitions information: all the information needed for the competitions' administration.

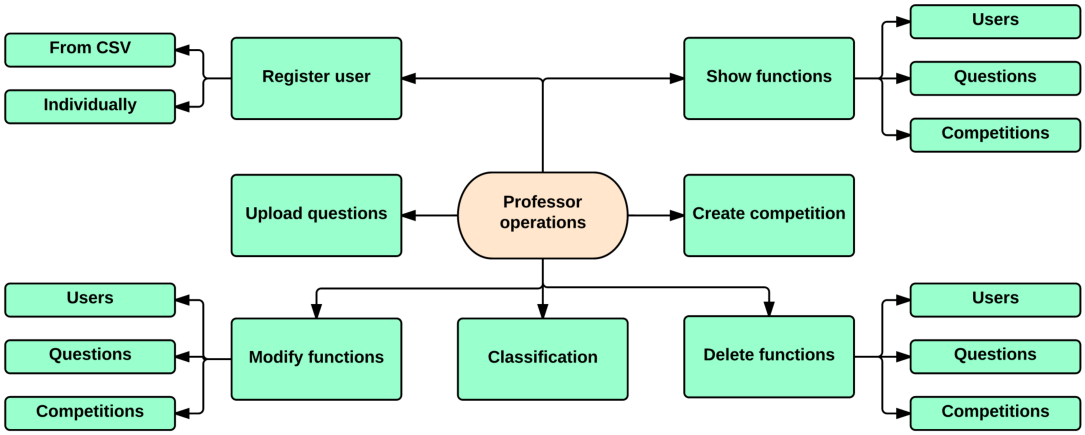


Figure: professor operations.

In the design and the implementation of the application the model-view-controller architectural pattern has been used.

Regarding at the data model, 3 fixed tables have been created in the database. These are:

- **Users table**: table that contain all the necessary parameters for the administration of the users: name, last name, identifier, E-Mail, groups, password, and privileges (to distinguish between professor and student).

- **Questions table**: table that contains all the questions information needed to create the tests: identifier, the topic the question belongs to, the level, the question and its 4 possible answers and the correct one.

- **Competitions table**: table that contains the data of the created competitions: name, included topics, group that participates in the competition, number of questions, initial date and final date, including time.

Apart from these 3 tables, 2 tables are created for each competition that the administrator creates. These two tables are always related to one row of the competitions table and get the information for its creation from the competition data and the users and questions tables. These 2 tables are:

- **Competition name questions table**: each of this table has the participant students as its rows and the question IDs as its columns. It is

the table where the questions that affect the competition are saved. The results will be saved here too.

- **Competition classification table**: when the competition has finished the result of the competition will be saved in this table.

The calculation of the final result of the test is simple. As each question has a level parameter associated to it, the result of a question will be the multiplication of each correct question times its level. Therefore, the result of a competition will be the sum of the result of all the questions contained in that competition.

In order to implement all the professor and student operations 37 PHP function were implemented. These functions get, update or delete information from the database tables.

Access control in the application is achieved by using session variables and two different privileges values: 0 and 1. Administrators of the program will have privileges 1, that is, access to all the administration operations and students will have privileges 0 that corresponds to students operations.

Login operation has been implemented using the E-Mail direction as username and asking for the password of the user to check with the one saved in the users table.

For all the operations listed previously error messages are displayed in the cases that the operation could not be done, the forms where left empty or the contents of them were incorrect or insufficient.

8.4. Validation

In the validation all the requirements of the application were validated. The requirements could be divided in two types: the ones fulfilled during design and development and the ones that needed concrete tests to validate them.

- Requirements achieved during design and implementation:
    - The independence and the contents of each table in the database: users, questions and competitions tables.
    - The existence of two different user types.

- Requirements that needed tests: the operations. The operations were tested by executing each of them in order to obtain the correct way. The tests were also made leaving the forms empty, inserting incorrect values manually in the server to test the inner functions of the server and trying to upload incorrect CSV files. Both Professor and student operations were tested.

At the end of the validation section it was proved that all the requirements were fulfilled by the application developed.

8.5. Conclusions and future work.

8.5.1. Conclusions.

Along the development of this project student has learnt about designing and implementing a web application based on client-server architecture.

It has been created a system that fulfil a list of requirements that was given by a professor of the university in order to work as a platform for creating competitions between students. All the functionalities needed by the administrator of a web application have been learnt.

The usage of CSV files for uploading the information give the application an easy way to upload data to the database and the versatility is obtained by having a wide range of operations available for the administrator, but it will be in a real environment where the system will show its good functioning.

It has been understood the necessity of conducting a theoretical analysis of the available technologies before starting a project, because it is the only way to get a general overview of the work that will be done.

Finally, the work that implies the design, development and the writing this thesis has given experience that the student did not have. Concepts learnt during the career have been reinforced and some others have acquired great value.

8.5.2. Future work.

The web application developed for this graduate project could be just the beginning of a bigger program that could be exported to mobile devices. Moreover, some ideas to improve and develop the application are:

- Creation of new competition modes: by creating question duels or a system with qualification rounds between the students the competitive environment could be increased.

- Including statistics functionalities: different types of classifications could be implemented: comparisons between students or progression graphs for analysing the progression.

- Creating a user-friendly graphical user interface for both students and professors.